UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE GRADO

EMULACIÓN EN FPGA DEL LAZO CERRADO DE UN CONTROLADOR DIGITAL PARA CONVERTIDOR CONMUTADO

Sandra Jurado Jabonero

Septiembre 2014

EMULACIÓN EN FPGA DEL LAZO CERRADO DE UN CONTROLADOR DIGITAL PARA CONVERTIDOR CONMUTADO

AUTORA: Sandra Jurado Jabonero

TUTOR: Ángel de Castro Martín

Trabajo realizado en el grupo

HCTLab

Human Computer Tecnology Laboratory

Escuela politécnica Superior

Universidad Autónoma de Madrid

Septiembre 2014

Agradecimientos

En primer lugar quiero agradecer a mi tutor, Ángel de Castro, todo el apoyo que me ha brindado en estos años. Desde "Sistemas de control", asignatura en la que coincidimos y sirvió para decidirme a hacer este trabajo, hasta ahora y durante la elaboración del mismo, ha sido un gran referente profesional para mí. Gracias por permitirme trabajar en su equipo y por intentar sacar lo mejor de mi durante este trabajo.

Gracias a mis compañeros de laboratorio por ayudarme cuando lo he necesitado, por enseñarme tantas cosas y por hacer más ameno el trabajo diario. Gracias por hacer que me sienta parte del grupo.

También quiero agradecer a mis compañeros su apoyo y su cariño durante estos años. Gracias por hacer de esta promoción un grupo tan especial en el que me he sentido muy a gusto. Muchas gracias por vuestras risas, vuestros consejos y vuestras enseñanzas. En especial quería agradecer a Laura su amistad durante estos años, sin ella hubiera sido más complicado llegar a este punto de mi vida. Gracias por compartir tanto tiempo inolvidable junto a mí y estar siempre a mi lado.

Por último, cómo no dar las gracias a mi familia, que ha soportado mis ausencias en las épocas de más trabajo, que han sido más extensas de lo que me hubiera gustado. A mis padres que han hecho posible que yo este hoy aquí, que me han educado para hacer las cosas lo mejor posible pero sin olvidar que en todo momento hay que ser *persona*. Por supuesto agradecer la ayuda incondicional de mi hermana, sus consejos, sus palabras y el estar ahí en todo momento, además de ser un referente en mí día a día. Además nombrar a Marcos que es parte de mi familia y que ha apoyado cada una de mis decisiones, ha disfrutado mis victorias y llorado mis derrotas tanto como yo.

A todos vosotros gracias por hacer esto posible.

Sandra Jurado Jabonero

Septiembre 2014

RESUMEN

En los últimos años se está produciendo un cambio en la electrónica que nos rodea. Los sistemas están pasando del mundo analógico al mundo digital en prácticamente todos los ámbitos en los que están presentes. En concreto, el uso de controles digitales ofrece beneficios importantes sobre el control analógico. El uso de sistemas digitales implica mejor reproducibilidad, facilidad de diseño, flexibilidad en la variación de los modelos y posibilidad de utilizar algoritmos más complejos entre otras ventajas.

Para llevar a cabo este diseño en el que una planta analógica se regula con un control digital es necesario realizar comprobaciones y verificaciones para cerciorar que el control a emplear funciona correctamente junto a la planta que manipula. En el caso de que existan fallos en este conjunto, como el uso de un control inapropiado, se puede traducir en el rediseño de la placa, en daños materiales o incluso en daños personales. Además para realizar esta comprobación en lazo cerrado, es decir, comprobar el control junto a la planta, sería necesario un sistema mixto que pueda manipular tanto señales digitales como analógicas. Las simulaciones mixtas requieren mucho tiempo de simulación y hay pocos simuladores disponibles en el mercado.

Para solventar este problema, se emplea la técnica HIL (Hardware In-The-Loop) que consiste en la digitalización de la planta y su integración en un dispositivo para lograr tiempos de simulación similares a los tiempos de trabajo del dispositivo analógico. Con esto se consigue simular 200 ms en un tiempo real de alrededor de 1 segundo, frente a las más de dos horas necesarias en los simuladores mixtos.

En este trabajo se trata este tema para el caso de los convertidores conmutados regulados con control digital. Para este caso se consigue la mejora de tiempos antes mencionada. Para ello se digitaliza el modelo de planta para así simularlo conjuntamente junto al control en lazo cerrado.

Por lo tanto, con estos resultados tomados de los modelos digitalizados que representan exactamente a las plantas sustituidas, se permite acelerar el proceso de pruebas y testeo de las fuentes conmutadas de una forma segura. Como se mostrará en este trabajo, el camino hasta lograr los objetivos marcados es algo complejo, pero con unos resultados asombrosos con respecto a las propuestas anteriores.

ABSTRACT

A change in the electronics surrounding us is taking place over the last few years. Analog systems are giving way to digital systems in most areas. Particularly, digital control systems provide some important advantages over analog ones, such as better reproducibility, easier designs, flexibility in model variation and more complex algorithms support.

In a design in which an analog plant is regulated with a digital controller, some verification is needed to check the correct work of that particular controller. If there is a failure such as inappropriate control, it may have implications like board redesign or personal and material damage. Moreover, in order to check in closed loop both the plant and the controller, a system that operates with analog and digital signals is required. Mixed simulations take a lot of time to run and there are few simulators available.

To overcome this problem, HIL (Hardware In-The-Loop) technique is proposed. It consists on digitizing the plant and integrating it in a device in order to achieve a performance similar to the one attainable with an analog system. HIL can run a simulation of 200 ms in approximately one second versus mixed systems which need more than two hours.

This work focuses on digitally-controlled switched converters. In order to obtain a significant time improvement in the verification of the closed loop system, a digitized model of the plant is generated so it can be easily simulated or emulated along with the controller.

Hence, the results of the digitized models represent accurately the substituted plants which leads to a faster and safer testing process. It will be shown that this is a complex task but its outcome is significantly better than previous works.

PALABRAS CLAVE

Convertidor elevador, fuente conmutada, planta analógica, controlador digital, emulación, simulación, hardware en lazo cerrado (HIL), digitalización, convertidor CC/CC, coma fija, resolución, precisión, pérdidas, recursos, simulador mixto.

KEYWORDS

Boost converter, switching source, analog plant, digital controller, emulation, simulation, Hardware-In-The-Loop (HIL), digitizing, DC/DC converter, fixed point, resolution, precision, losses, resources, mixed simulator.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE ABREVIATURAS

ADC (Analog-to-Digital Converter)

ALU (Unidad Aritmético Lógica)

ASIC (Application Specific Integrated Circuits)

CC (Corriente Continua) o DC (Direct Current)

CCM (Continuous Conduction Mode)

CFP (Corrección del Factor de Potencia) o PFC (*Power Factor Correction*)

DCM (Discontinuous Conduction Mode)

DSP (Digital Signal Processor)

ESR (Equivalent Series Resistance)

FF (Flip Flop)

FPGA (Field-Programmable Gate Array)

HDL (Lenguaje de Descripción de Hardware)

HIL (Hardware In-the-Loop)

LUT (Look Up Table)

MOSFET (Metal-oxide-semiconductor Field-effect transistor)

PID (Proporcional Integral derivativo)

PWM (Pulse-Width Modulation)

VHDL (VHSIC Hardware Description Language)

VHSIC Very-High-Speed Integrated Circuits

1 INTRODUCCIÓN

Las fuentes de alimentación tienen una gran importancia actualmente debido a que están presentes en todos los dispositivos electrónicos que nos rodean, desde teléfonos móviles, televisiones, vehículos, etc. Un esquema básico de fuente de alimentación que transforma la corriente alterna (CA) en corriente continua (CC) se muestra en la Figura 1. En ella se puede observar cómo la corriente alterna que entra en el circuito pasa por un transformador en el que se reduce el valor medio de la tensión al valor deseado, después pasa por un puente de diodos que rectifica la señal. A continuación, la señal pasa por un filtro para conseguir una tensión aproximadamente continua. Por último, se requiere el uso de un convertidor de corriente continua a continua para reducir el rizado.

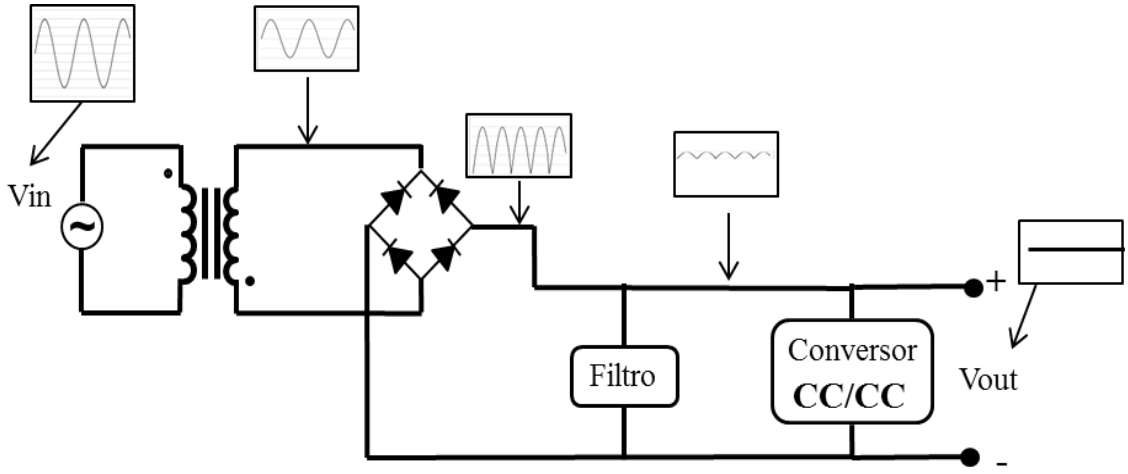

Figura 1. Modelo completo de una fuente de alimentación

El desarrollo de este trabajo se centra en el convertidor CC/CC que se puede ver en la fuente CA/CC. En el mercado existen numerosos convertidores CC/CC, desde uno de los más sencillos como es el regulador lineal, hasta otros más complejos como los convertidores conmutados. Estos últimos serán concretamente los empleados para realizar este trabajo.

El regulador lineal cuyo esquema se muestra en la Figura 2 tiene un funcionamiento muy sencillo que básicamente consiste en disipar la energía suficiente para obtener el valor de tensión deseado a la salida. Para esto se emplea una tensión de referencia que se compara con la salida y se varía el estado de los conmutadores en función de esta diferencia para aumentar o disminuir la tensión de salida.

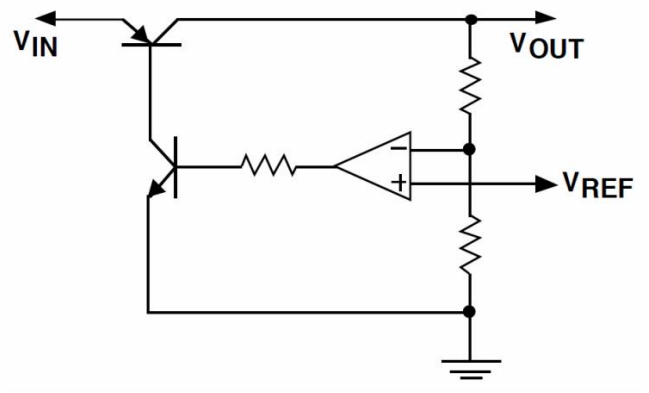

Figura 2. Esquema de un regulador lineal

Por otro lado, los convertidores conmutados están formados por elementos no disipativos, por lo que la eficiencia es muy alta, mucho mayor que en el caso del regulador lineal, ya que no sufren pérdidas resistivas. Además estos convertidores tienen varios conmutadores, como diodos o transistores, que permiten el paso de la corriente y con ello modificar la tensión o corriente de salida. Los convertidores son acompañados por un sistema de control que manipula la apertura y cierre de los conmutadores en función de la salida deseada. Estos convertidores pueden ser de varios tipos dependiendo de la disposición de sus elementos se puede diferenciar entre los convertidores reductores (conocidos por su nombre en inglés, *buck*) o los convertidores elevadores (más conocidos por su nombre en inglés, *boost*) entre otros. Este último será objeto de estudio en este trabajo.

Este sistema conjunto del convertidor conmutado y el control se ha resuelto típicamente con técnicas analógicas, es decir, construyendo el sistema conjunto y verificándolo con herramientas apropiadas como osciloscopios y analizadores para comprobar su correcto funcionamiento. Esta técnica, pese a que está muy arraigada, tiene varios problemas. Entre ellos se encuentran los daños materiales que puede causar un error en el diseño del control, que implica unos daños económicos e incluso personales. Además es necesario reconstruir el circuito para variar el sistema de control. Por este motivo se busca otra forma de analizar este regulador en su periodo de pruebas.

Otra de las técnicas que se valora para este caso es realizar una simulación mixta en la que se mantiene el modelo del *boost* de planta como un dispositivo analógico mientras que el regulador se pasa al mundo digital. Es necesario verificar el control junto a la planta que regula para evitar errores de diseño. Para realizar esta simulación mixta del controlador digital junto al convertidor conmutado analógico se requieren unos conversores de señal (DAC, Digital to Analog Converter para pasar del mundo digital al aalógico y ADC, Analog to Digital Converter para el proceso opuesto) como se muestra en la Figura 3. Para abordar esta verificación existen varias opciones como se verá en el estado del arte, desde simuladores mixtos con los que probar el control junto al convertidor, pero son muy lentos y además solo están disponibles en versiones de pago,

hasta modelos aproximados que describen la planta y se prueba junto al control en un sistema de simulación como Matlab-Simulink. Esta solución no es la ideal ya que se deben probar los modelos finales para que no haya diferencias entre los resultados obtenidos y el modelo realmente diseñado.

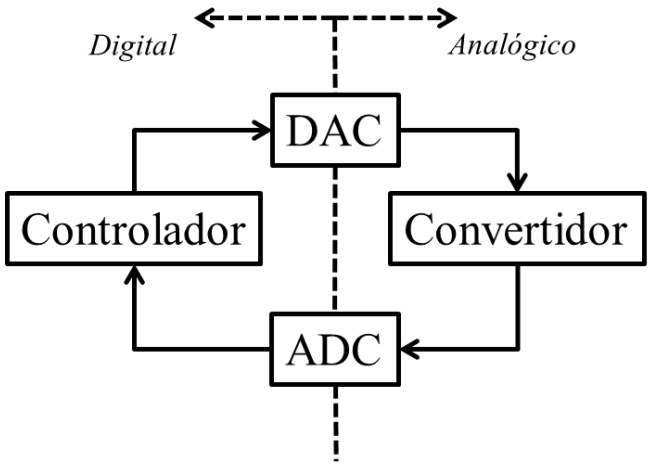

Figura 3. Sistema mixto de un convertidor de potencia con control digital

Debido a los problemas de las opciones anteriores para la simulación, en este trabajo se baraja una alternativa para conseguir modelos prácticamente idénticos a los empleados en el convertidor analógico pero a su vez con simulaciones muy rápidas. Esto se consigue llevando el modelo del convertidor al mundo digital y simulándolo junto al control, ambos definidos en el mismo lenguaje. Además, si estos modelos de control y convertidor se describen en un lenguaje como VHDL (VHSIC *Hardware Description Language*) se pueden introducir los diseños en un dispositivo como FPGA y emularlos, es decir ejecutarlos en el dispositivo programable conjuntamente consiguiendo un tiempo ligeramente superior al que se quiera emular, y mucho menor al necesario en los casos anteriormente descritos.

Esta nueva alternativa de verificación se conoce como HIL (*Hardware In-the-Loop*) y consiste en realizar las simulaciones en lazo cerrado, es decir verificando el control digital junto al modelo de planta que regula, ambos implementados sobre hardware digital. Aunque en este caso estos modelos se correspondan con los empleados para el diseño de una fuente de alimentación, esta técnica se emplea para verificar cualquier tipo de controlador digital junto a la planta sobre la que actúa y detectar así cualquier error antes de su puesta en marcha.

Por lo tanto, el objetivo de este trabajo consiste en implementar un modelo de planta que se corresponda con el de un convertidor elevador en VHDL y cuya verificación junto a su control sea más rápida que la que se consigue con otras formas de implementación para su comprobación.

Para ello se realizará un modelo ideal en VHDL del esquemático y posteriormente un modelo teniendo en cuenta las pérdidas del circuito analógico en VHDL. Estos modelo tienen unas señales que se deben definir con un tipo de datos numérico. En un primer caso se emplea el tipo de datos *real* con el que se consigue una resolución muy buena ya que se trata de un tipo de datos en coma flotante con 64 bits de resolución. El problema de este tipo es que no se puede implementar en FPGA, por lo tanto se realiza un segundo diseño con las señales definidas en *sfixed*. Este formato forma parte de la librería de VHDL y se encarga de definir señales en coma fija como se explicará más adelante. Una de sus principales características es que es soportado por FPGA y por lo tanto se pueden emular los diseños que contengan este tipo de datos. Por otro lado, el número de bits de resolución que se utiliza es mucho menor que en el resto de casos y los tamaños de las señales los debe decidir el diseñador.

Tras la elaboración de estos cuatro diseños descritos anteriormente que tendrán lugar en el tercer apartado, se pasa al análisis de resultados. En primer lugar se analizará la resolución de las señales de los diseños llevados a cabo en coma fija. Posteriormente se analizarán los recursos empleados por los diseños con mejor resolución y por último se tomarán los tiempos de simulación y emulación de cada uno de los diseños y se comparará con los tiempos necesarios para la simulación con modelos mixtos. Con esto se conseguirá una comparativa entre la propuesta HIL contra la simulación mixta, concluyendo que esta última es mucho más lenta.

2 ESTADO DEL ARTE

El objetivo de las fuentes de alimentación es adaptar los requisitos de tensión y corriente de la carga al generador. Para ello se pueden emplear los convertidores que, como su nombre indica, convierten una forma de onda de cierto tipo o nivel en otra. Como se muestra en [8], este cambio puede ser de un valor a otro o de un tipo a otro, con tipo se refiere a una transformación de corriente continua (CC) a corriente alterna (CA) o viceversa.

Existen dos tipos básicos de fuentes de alimentación: los reguladores lineales y los convertidores conmutados. Los reguladores lineales basan su funcionamiento en la disipación de energía por medio de una resistencia variable, mientras que los convertidores conmutados se basan en el uso de transistores para permitir o negar el paso de energía y así hacer que varíe a su salida según dictan unas señales de control.

Los reguladores lineales tienen muy poca eficiencia ya que su funcionamiento consiste en disipar el excedente de energía en forma de calor, por lo que además sufren un calentamiento mucho más alto que el sufrido en los convertidores conmutados y no se consideran una buena opción para potencias de salida mayores de 40 vatios puesto que se disiparía mucha energía. Por otro lado, mientras el regulador lineal solo permite disminuir el valor de la señal de salida respecto al de entrada, los convertidores conmutados pueden aumentar o reducir el valor de salida respecto a la entrada. Además, los reguladores lineales se utilizan con transformadores de bajas frecuencias, típicamente más grandes que los empleados en una fuente conmutada como se ve en [6].

Pese a las ventajas del convertidor conmutado sobre el regulador lineal, también existen varias desventajas, entre otras que el regulador lineal es más sencillo debido a que el convertidor conmutado incluye una etapa de control más compleja. Además debido a los cambios abruptos de corriente que se dan en el convertidor conmutado, se pueden generar radiaciones indeseadas que es necesario filtrar.

Conociendo las opciones, en este caso se trabajará con los convertidores conmutados pues es la opción mayoritaria para fuentes de decenas de vatios en adelante. Como ya se ha mencionado existen varios tipos, como el convertidor reductor (conocido por su nombre en inglés *buck*), el convertidor elevador (en inglés *boost*) y el convertidor buck-boost, que como su nombre indica puede actuar de reductor o de elevador. A lo largo de este trabajo se empleará el convertidor elevador o *boost* por ser el más habitual en CA/CC con CFP (Corrección del Factor de Potencia).

Como se ha mencionado anteriormente, los convertidores conmutados deben ir acompañados de un sistema de control que genere una señal modulada por ancho de

pulso que se inyecte al transistor. Este sistema de control busca controlar la tensión que se entrega a la salida. En el caso que muestra la Figura 4 el control se realiza con un lazo lento que muestrea la tensión de salida y la compara con la de referencia deseada con un muestreo a 100 Hz. Con este error y mediante un mecanismo de control por realimentación (típicamente PID, Proporcional Integral Derivativo) se calcula la desviación del error como la conductancia de entrada, es decir, el inverso de la resistencia que debe ver como entrada la red eléctrica. Dicha conductancia multiplicada por la tensión de entrada medida establece la referencia de la corriente de entrada. Este paso es necesario ya que se busca controlar también la corriente de entrada para que sea proporcional a la tensión de entrada y que no haya armónicos, es decir, evitar los picos de corriente indeseados. Con este valor se halla la corriente de referencia a la entrada para así compararla con la corriente de entrada actual en un lazo con una frecuencia de muestreo de 100 kHz. Con la diferencia entre la corriente de referencia y la corriente de entrada se genera una señal modulada por ancho de pulso (PWM, Pulse-Width Modulation), registrada a una frecuencia de reloj de 100 MHz, que regula la apertura y cierre del transistor.

Por lo tanto, para cada periodo de muestreo del lazo lento, que compara la tensión de salida con una de referencia, son necesarios 1000 periodos de muestreo del lazo rápido en el que se compara la corriente de entrada con la de referencia, que a su vez genera la señal PWM que requiere 1000 ciclos de reloj. Por lo tanto para cada periodo de muestreo del lazo lento son necesarios 1 000 000 de ciclos de reloj. Si además se quieren ver varios periodos del lazo lento, se necesitan millones de ciclos de reloj que llevan mucho tiempo simular.

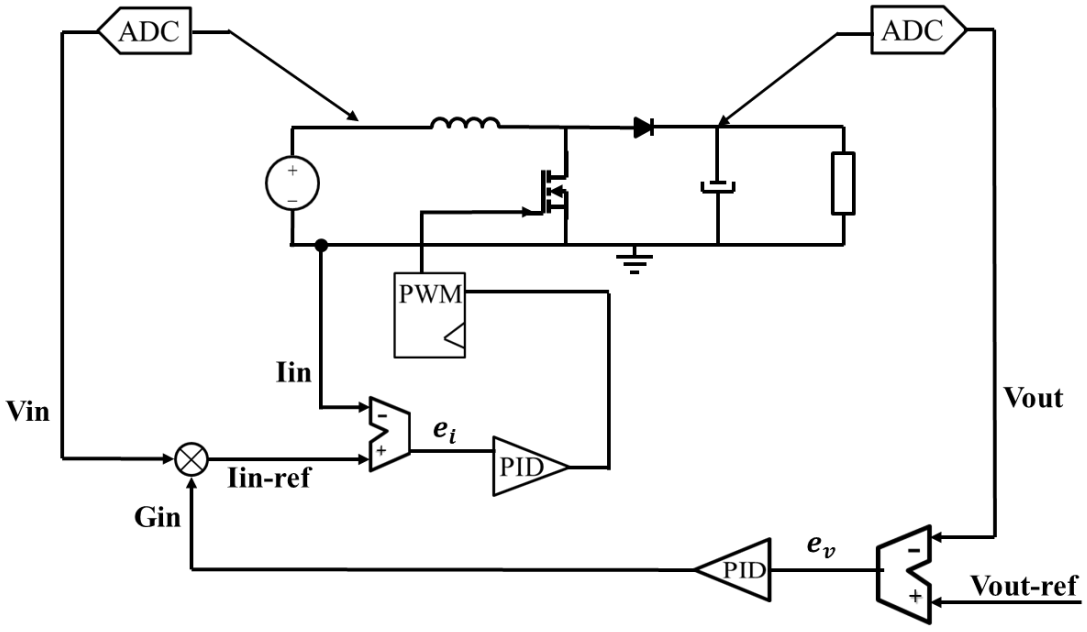

Figura 4. Fuente conmutada junto al sistema de control

Con la intención de manipular este control como se ha comentado anteriormente, podemos fijarnos en lo explicado en el capítulo 6 de [8]. Para el circuito del convertidor conmutado elevador de la Figura 7 se puede conseguir una tensión de salida u otra en función del tiempo en que permanezcan abiertos o cerrados los conmutadores. Siendo D el ciclo de trabajo de una señal (Duty cycle en inglés), es decir, la relación entre el tiempo que permanece activa y el periodo propio de la señal, se pueden aproximar los valores de salida en función de este valor.

Para el modelo del convertidor elevador cuando el transistor se encuentra cerrado, la ecuación que rige la bobina se puede expresar como:

$$\frac{V_g}{L} = \frac{di_L}{dt} \cong \frac{\Delta i_L}{\Delta t} = \frac{\Delta i_L}{Dt} \tag{2.1}$$

Por lo que la ecuación de una de las variables de estado del convertidor elevador, se puede aproximar como

$$(\Delta i_L) = \frac{V_g DT}{L} \tag{2.2}$$

Y siguiendo el mismo planteamiento para el transistor abierto, se puede modelar con las siguientes ecuaciones:

$$\frac{V_g - V_0}{L} = \frac{di_L}{dt} \cong \frac{\Delta i_L}{\Delta t} = \frac{\Delta i_L}{(1-D)t} \tag{2.3}$$

Obteniendo como resultado para la variable de estado:

$$(\Delta i_L) = \frac{(V_g - V_0)(1-D)T}{L} \tag{2.4}$$

Estas ecuaciones anteriores se muestran en la Figura 5 en función del tiempo para cada periodo de conmutación. Y como se muestra con las ecuaciones (2.2) y (2.4), despejando el ciclo de trabajo se puede regular el pulso de apertura y cierre del transistor y así obtener la tensión deseada a la salida:

$$\frac{\mathrm{V_0}}{\mathrm{V_g}} = \frac{1}{(1-\mathrm{D})} \tag{2.4}$$

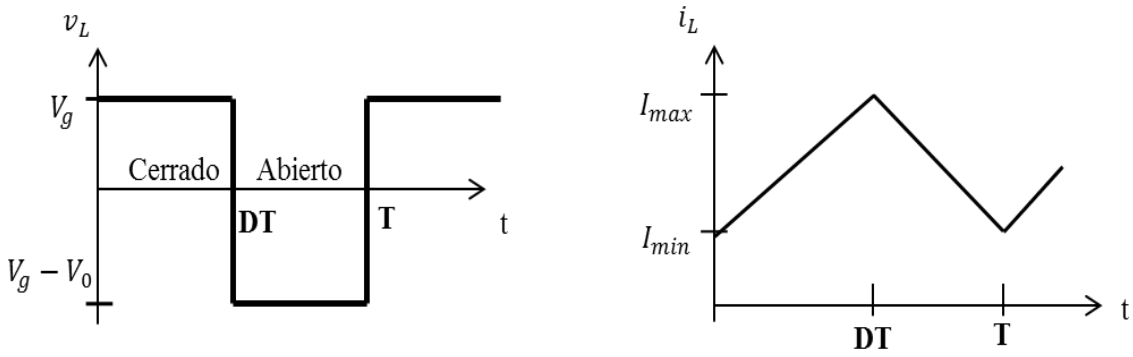

Figura 5. Tensión y corriente en la bobina según el estado del transistor

Para diseñar fuentes conmutadas es necesario realizar simulaciones del funcionamiento completo del convertidor conmutado junto al sistema de control que lo manipula. Como se muestra en [2] la tradición para los sistemas de control en convertidores conmutados ha sido analógica. Debido a los avances en cuanto al control digital, esta solución se podría implantar como mayoritaria en los próximos años, pero es necesario un periodo de investigación y pruebas previo.

Los motivos para pasar del mundo analógico al digital son varios como la capacidad de reprogramación, la posibilidad de introducir algoritmos más complejos, la capacidad de monitorización, la disminución del tiempo de diseño, aumenta la fiabilidad del sistema y se consigue una mayor integración. Además, como ya se sabe los sistemas digitales no son tan sensibles a cambios propios del paso del tiempo y son menos sensibles al ruido puesto que trabajan en dos valores (1 o 0).

Para dar este salto, también hay que ser consciente de los inconvenientes que supone este cambio. Para empezar se debe intercomunicar el mundo analógico de la planta con el mundo digital del controlador por lo que son necesarios convertidores analógico/digitales (CAD y ADC), el uso de estos elementos junto al tiempo de cálculo influye en el retardo del control. Además como se abordará en este trabajo, hay que tener en cuenta las limitaciones de resolución debidas a que los valores se representan con un valor fijo de bits. Otro de los inconvenientes que ralentiza la expansión del control digital, aparte del aumento de precio que supone, consiste en el desconocimiento de los diseñadores de control sobre estas nuevas técnicas.

Sopesando ventajas frente a inconvenientes se puede determinar que dependiendo del uso de cada dispositivo se aconseja el empleo de la técnica de control más apropiada, ya sea digital o analógica.

Este trabajo se centra en el control digital y concretamente en uno de los obstáculos que supone el hecho de realizar pruebas con un control digital y una planta analógica, que pese a que en este caso se trate de un convertidor conmutado y el control de esta

fuente, estas técnicas se pueden expandir al control de cualquier planta analógica con control digital.

Existen numerosos estudios sobre las distintas posibilidades de simulación que se ofertan para sistemas mixtos. En [11] se realizan varias simulaciones, en la primera se emplea un sistema con Matlab/Simulink que permite una simulación rápida y un sistema de verificación para resoluciones y retardos, pero no se corresponde completamente con el modelo del sistema a verificar. En la segunda simulación se emplea un simulador mixto, Cadence Spectre, para un convertidor analógico definido en Spice detalladamente y un controlador descrito en HDL (Hardware Description Language), para este caso se requiere mucho tiempo de simulación. La última propuesta es igual que la anterior pero empleando modelos genéricos de Spice, lo que permite mayor aceleración en las simulaciones. De igual modo estas simulaciones mixtas necesitan varias horas por ciclo de conmutación.

Ya que existen pocos simuladores mixtos, existen otras propuestas que emplean dos simuladores [7], uno analógico, PSIM, y otro digital, Modelsim. El problema es que para llevar a cabo estas simulaciones conjuntamente, es necesario que el diseñador cree una interfaz para enlazar ambos programas. Para esto puede emplear un lenguaje como C o C++ y se debe tener en cuenta la sincronización de datos, los flujos de realimentación, etc.

Otra de las modalidades de simulación consiste en modelar la planta en HDL (Lenguaje de Descripción de Hardware) a partir de las ecuaciones en diferencias que modelan el circuito. Así en [4] se compara el uso de Spice, VHDL y VHDL-AMS (consiste en una extensión de VHDL para definir señales digitales y analógicas). Donde se concreta que empleando Spice como referencia, VHDL-AMS es 2,18 veces más rápido y la simulación con VHDL es 3,18 veces más rápida. Esta comparación entre VHDL-AMS y VHDL también se comprueba en [5] donde se consiguen tiempos de simulación 10 veces mayores para VHDL que para VHDL-AMS.

En general las simulaciones para el modelo del convertidor digital son más rápidas que las simulaciones mixtas que como se muestra en [3] para simular 200 ms son necesarias 2 horas 13 minutos y 21,751 milisegundos. Pero en ocasiones no son lo suficientemente rápidas como para llevar a cabo simulaciones de muchos ciclos de conmutación. Para estas simulaciones largas existe una solución conocida como HIL (Hardware In-The-Loop) que se empleará en este trabajo.

Un sistema HIL consiste en realizar una implementación digital de los modelos descritos en un ordenador o en una FPGA, para poder simular todo el conjunto de control y planta en lazo cerrado. En este caso se emplea un modelo HIL sobre FPGA, ya que se consiguen velocidades mayores que sobre ordenador. Además se permite simular los modelos finales, lo cual es muy importante para evitar errores de verificación que pueden llevar a modelos erróneos. El problema de esta verificación reside en el empleo

del tipo de datos soportado por la mayoría de FPGAs, coma fija. Ya que coma flotante no es sintetizable por muchas FPGAs además implica tiempos de simulación mayores que en coma fija para errores similares o mayores respecto a las simulaciones reales.

Para llevar a cabo el modelo del convertidor en FPGA es necesario pasar el diseño del circuito del *boost* a VHDL. Esto se consigue, como se ha dicho anteriormente, pasando las ecuaciones diferenciales que lo definen a ecuaciones en diferencia como se verá en el apartado 3.

Tras definir el modelo matemático, es necesario pasar el diseño a VHDL. Para ello existen tres opciones para definir las señales numéricas:

- *Real*: Este tipo de datos consiste en un modelo en coma flotante, es decir un tipo de notación científica binaria de 64 bits de resolución. Esto permite resultados de precisión prácticamente inequívocos con 52 bits para la mantisa y 11 para la representación del exponente. El principal problema de este tipo de datos reside en que no es posible su emulación en FPGA ya que estos dispositivos no soportan este formato de datos. Pese a que no será el modelo que permita realizar el sistema HIL en FPGA será empleado como modelo patrón para los diseños en coma fija.

- Coma flotante: Este tipo de datos es emulable solamente por algunas FPGAs. No requiere un esfuerzo extra por el diseñador para seleccionar los tamaños oportunos para las señales a definir. Pero este modelo no se baraja entre las opciones de diseño debido a que con un tipo de coma flotante de 32 bits no se obtiene resolución suficiente para definir algunas señales. Además requiere tiempos de simulación mayores que coma fija.

- Coma fija: Este tipo de datos es emulable y simulable, requiere un trabajo por parte del diseñador para definir los tamaños oportunos para cada señal, pero como parte positiva, mejora los tiempos necesarios para la verificación del convertidor junto al control.

El tipo numérico de coma fija consiste en definir un número concreto de bits para la parte entera del número (X) y para los fraccionarios (Y), además el bit más significativo representa el signo del número en complemento a2:

$$-2^{X} + 2^{X-1} + 2^{X-2} \dots + 2^{2} + 2^{1} + 2^{0} + 2^{-1} + 2^{-2} \dots + 2^{Y-1} + 2^{-Y}$$

Una forma rápida de representar estos formatos es la nomenclatura QX.Y, donde X representa el número de bits de la parte entera e Y hace alusión al número de bits de la parte decimal. Por lo tanto para el número 11111 en formato Q3.1, el número que se representa es $-2^3 + 2^2 + 2^1 + 2^0 + 2^{-1} = -0{,}5$ en decimal. Por otro lado este

mismo número binario en formato Q2.2, representa el número decimal $-2^2 + 2^1 + 2^0 + 2^{-1} + 2^{-2} = -0{,}25$.

Para definir coma fija en VHDL se emplea una biblioteca propia de este lenguaje diseñada para VHDL-2008. Esta biblioteca permite definir números gracias al tipo de datos sfixed que permite especificar el número de bits para la parte entera y el número de bits para la parte decimal para números con o sin signo. Así para definir una señal de tipo Q2.3 en VHDL se debería declarar como: "sfixed (2 downto -3)".

La validez de estos diseños propuestos a desarrollar se muestra en [1], de donde se ha conseguido la Figura 6 que muestra varios valores para varias propuestas en las que se simula el modelo de un convertidor elevador. Como se ve en esta gráfica, un modelo en coma flotante de 32 bits es el que más difiere de los resultados experimentales, que son los resultados reales del convertidor analógico. Esto se debe a que coma flotante de 32 bits no es suficiente para este tipo de modelos. La simulación más parecida a los datos experimentales es la simulación mixta, mientras que las simulaciones tanto con *real* como con coma fija son muy similares entre sí y se aproximan enormemente a la simulación mixta. En este trabajo nos centraremos en los modelos *real* y en coma fija, pero añadiendo también pérdidas a dichos modelos, de tal forma que sean más parecidos a la realidad que los modelos planteados en [1], que son siempre modelos sin pérdidas.

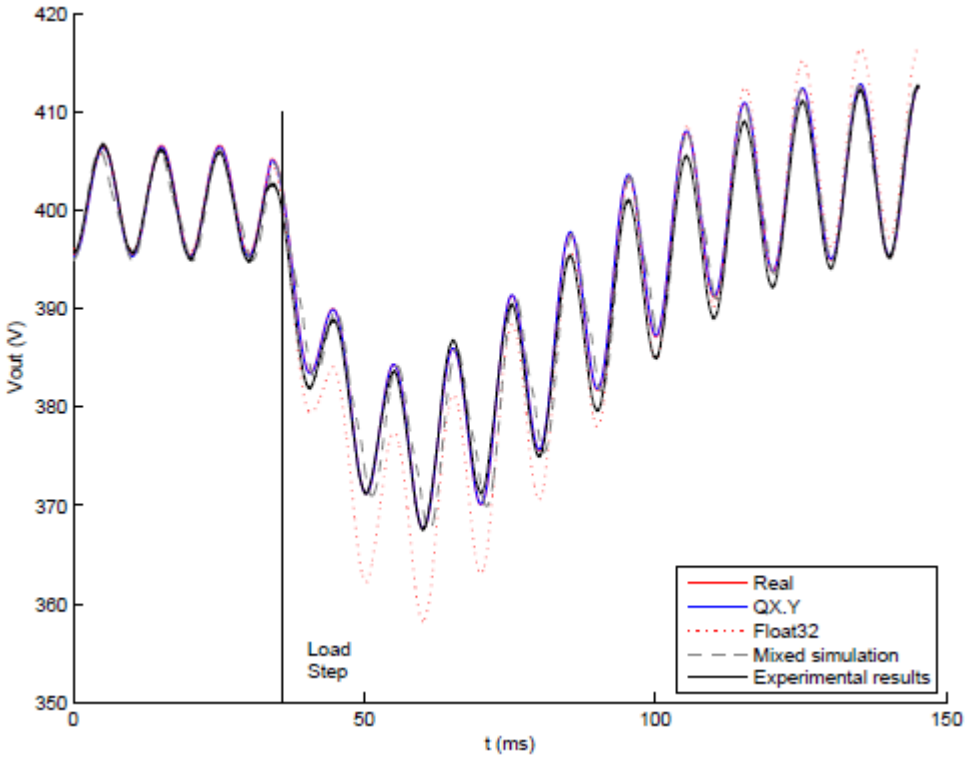

Figura 6. Comparación de modelos propuestos para simulación

Por último, una vez realizados los modelos en coma fija, se busca la forma más idónea de tomar los datos de salida de la FPGA con los que realizar la verificación pertinente.

Para eso se puede utilizar un osciloscopio con el que medir las señales de salida de la FPGA. Pero en este caso se ha empleado una herramienta llamada ChipScope de Xilinx.

Este programa, que se ejecuta en un ordenador, permite mediante un cable JTAG tomar los valores de las entradas y salidas de una FPGA para su análisis mediante el monitor del ordenador. Su interfaz permite agrupar señales para formar buses con los que ver señales de varios bits simultáneamente y permite añadir disparos como en un osciloscopio para seleccionar el momento en el que se empieza a capturar datos o qué datos almacenar.

Por lo tanto en este trabajo se han barajado varias opciones entre las que se ha seleccionado realizar el diseño de un convertidor conmutado con control digital que será simulado tras digitalizar el diseño del convertidor. Además serán diseñados en un modelo real que servirá de patrón para posteriormente poder realizar la emulación en coma fija sobre FPGA y así analizar los resultados mediante ChipScope. Con todo esto se consiguen verificaciones rápidas que requieren poco tiempo extra por cada ciclo de conmutación y que representan los modelos reales que se quieren comprobar.

3 DISEÑO DE MODELOS VHDL DE CONVERTIDORES CONMUTADO

Como se ha visto anteriormente, existen muchas fuentes de alimentación de diversos tipos. A lo largo de este capítulo nos centraremos en los convertidores conmutados, concretamente, en los convertidores conmutados con control digital. El objetivo es la verificación del lazo cerrado de un convertidor conmutado, típicamente analógico, con un control digital. Como se ha visto anteriormente, se opta por un modelo en VHDL del modelo del convertidor para su verificación con el regulador definido de la misma manera.

Este modelo del convertidor será, por lo tanto, verificado en lazo cerrado, y se le aplicará corrección del factor de potencia o PFC (*Power Factor Correction* en inglés). En este caso se controla tanto la tensión de salida para que se ajuste a una de referencia, como la corriente de entrada para que sea proporcional a la tensión de entrada en el lazo cerrado. El problema del lazo cerrado es que se pueden esconder algunos errores del diseño debido a que el control lleva tanto la corriente como la tensión a su valor de consigna, aunque el modelo no sea el adecuado. Así pues, el convertidor será verificado en lazo abierto previamente. En este caso se analizará el modelo en lazo abierto haciendo hincapié en que las resoluciones empleadas para el modelo en coma fija sean adecuadas y las ecuaciones no sean erróneas.

El modelo de planta que se va a implementar para realizar las verificaciones pertinentes a lo largo de este capítulo es un convertidor elevador (también conocido por su nombre en inglés: *boost converter*). Este circuito es un convertidor de corriente continua a continua (CC/CC, o en sus siglas en inglés DC/DC, *Direct Current*), cuya tensión a la salida es mayor que la tensión a su entrada. Se emplea este diseño debido a que es el más común en corrección del factor de potencia.

3.1. Modelo de un convertidor elevador

Un convertidor *boost* está compuesto por dos interruptores (típicamente un transistor y un diodo) que permiten aumentar la tensión que llega a la entrada del circuito, además de un condensador y una bobina que regulan el rizado de la señal de salida. El hecho de que este diseño no contenga resistencias, a excepción de la carga a la salida, hace que no sufra las pérdidas propias del uso de este elemento y así se consigue aumentar la eficiencia de la planta debido a que la energía entregada al circuito no se disipa sino que se transforma.

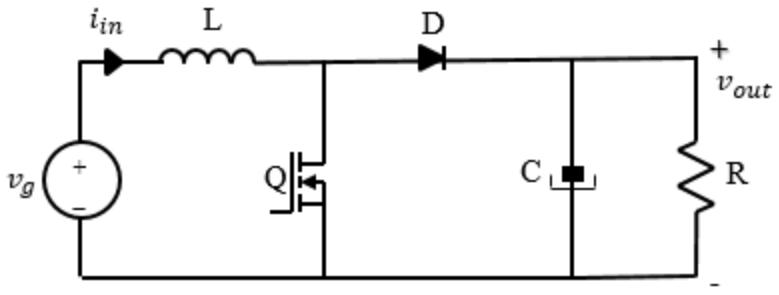

Figura 7. Esquema eléctrico del convertidor elevador

En este trabajo se busca realizar un modelo del *boost* para llevar a cabo una simulación y verificación rápida para fuentes conmutadas. Por lo que se va a desarrollar el convertidor conmutado mostrado en la Figura 7 en VHDL. La elección de VHDL para implementar este modelo tiene que ver con el hecho de que el regulador sea digital, y así se consigue tener el modelo completo (regulador y planta) en el mismo lenguaje y facilitar su verificación. Para conseguir esto, se debe analizar el circuito de la Figura 7, discretizando las ecuaciones diferenciales de sus variables de estado, es decir, la corriente que circula por la bobina y la tensión en el condensador.

En primer lugar, se observa que la corriente que atraviesa la bobina de valor L (i_L), coincide con la corriente a la entrada del convertidor (i_{in}). La tensión que cae en la bobina (v_L) se puede definir como:

$$v_L(\mathrm{t}) = L \cdot \frac{di_L(t)}{dt} \tag{3.1.1}$$

En este caso, se aproximan las derivadas e integrales que aparezcan como ecuaciones en diferencias, y así la expresión anterior se puede reescribir despejando la corriente que atraviesa la bobina como:

$$i_L(k) = i_L(\mathrm{k}-1) + v_L(\mathrm{k}) \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}} \tag{3.1.2}$$

Siendo Δt el paso de integración para calcular las variables de estado, k el instante de tiempo en el que se evalúa la ecuación, y por lo tanto k-1 el instante anterior al instante k.

Siguiendo el mismo planteamiento visto en la bobina para el condensador, se puede describir la corriente que lo atraviesa (i_C) en cada momento como:

$$i_C(\mathrm{t}) = C \cdot \frac{dv_C(t)}{dt} \tag{3.1.3}$$

De la misma manera antes vista para la bobina, desarrollando la ecuación anterior, se consigue la ecuación en diferencias correspondiente a la tensión de salida, que coincide con la que cae en el condensador (v_C) en un instante de tiempo k:

$$v_C(k) = v_C(\mathrm{k}-1) + i_\mathrm{C}(\mathrm{k}) \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}} \tag{3.1.4}$$

Por lo tanto, se obtienen las ecuaciones que definen las variables de estado del convertidor conmutado, es decir, la tensión que cae en el condensador (v_C) y la corriente que atraviesa la bobina (i_L), que a su vez, como ya se ha comentado anteriormente, coinciden respectivamente con la tensión de carga que ofrece el *boost* a su salida (v_{out}) y con la corriente que entra a éste (i_{in}).

Para llevar estas fórmulas a VHDL, se define $\Delta \mathrm{t}$ como un tiempo constante, por lo que los términos de las ecuaciones anteriores $\frac{\Delta \mathrm{t}}{\mathrm{C}}$ e $\frac{\Delta \mathrm{t}}{\mathrm{L}}$ son constantes, es decir, van a tener un valor fijo independientemente del instante de tiempo k en el que se encuentre. Además los términos $i_\mathrm{C}(\mathrm{k}-1) \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}}$ y $v_L(\mathrm{k}) \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}}$ son los incrementos que se añaden en cada iteración a las variables de estado en el instante anterior.

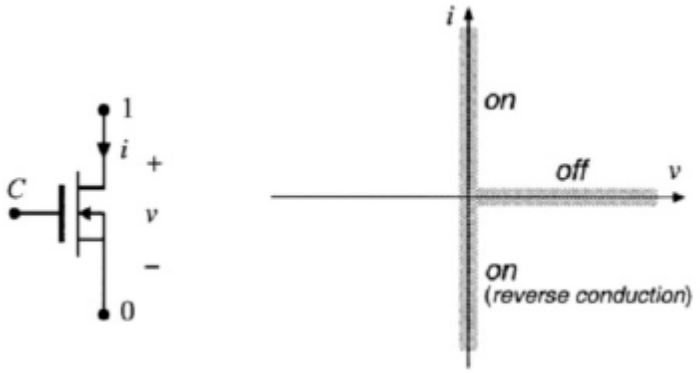

Figura 8. Transistor MOSFET ideal

Por otro lado, el *boost* también posee un transistor y un diodo que modifican el estado del circuito. Cuando el transistor, que en este caso será un transistor tipo *mosfet* ideal como el que muestra la Figura 8, permite el paso de corriente, toda la corriente que entra al circuito pasa por él, por lo que la corriente que atraviesa el diodo es nula, este caso se puede visualizar en la Figura 9. Así la corriente que atraviesa el condensador (i_C) es igual a la corriente que atraviesa la resistencia de carga a la salida (i_R) y la tensión en la bobina (v_L) tiene el mismo valor que la tensión de entrada (v_g).

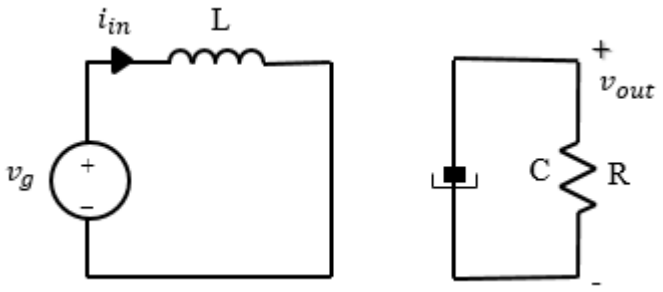

Figura 9. Esquema eléctrico del *boost* con mosfet on

Cuando el transistor no permite el paso de corriente, la corriente que atraviesa la bobina pasa directamente por el diodo. Pero el diodo puede estar en dos posibles estados: conduciendo (CCM, o en inglés *Continuous Current Mode*) cuando la corriente atraviesa el diodo desde su ánodo hasta su cátodo o no conduciendo (DCM, del inglés *Discontinuous Current Mode*) que se refiere al estado en el que el diodo no conduce, cuando este es ideal como se ve en la Figura 10. Si el diodo no es ideal, tiene otro estado de saturación, en el que puede circular una pequeña corriente desde su cátodo hasta su ánodo. Para este análisis, el caso de saturación será tratado como un DCM. Así el funcionamiento del diodo se modela únicamente como el que se muestra en la Figura 10.

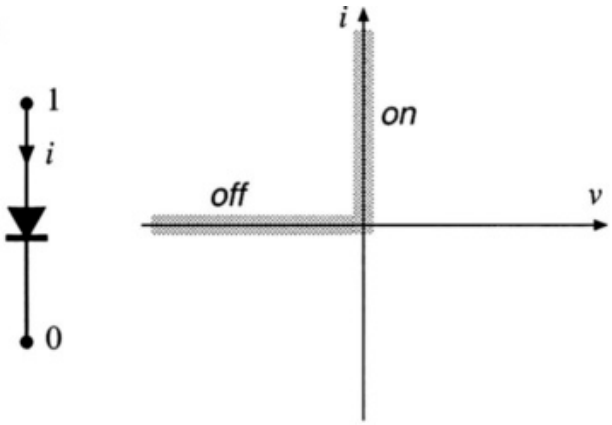

Figura 10. Diodo ideal

Como se explica en el párrafo anterior, uno de los posibles estados del circuito del convertidor es que este el *mosfet* sin conducir y el diodo en modo CCM, como se muestra en la Figura 11 obtenida de [10], donde la corriente que atraviesa el *mosfet* es nula y así, la corriente que entra al circuito circula íntegramente por el diodo. Por lo tanto, i_C en este caso, será $(i_L - i_R)$ mientras que el valor de v_L es $(v_g - v_o)$.

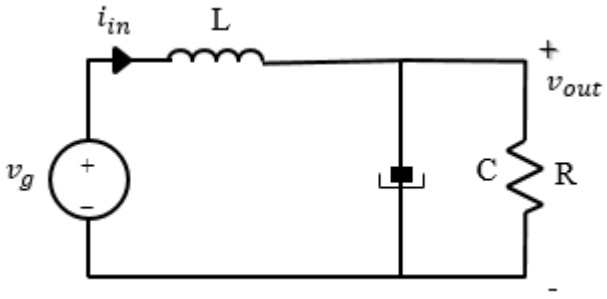

Figura 11. Esquema eléctrico del *boost* con mosfet off y diodo en CCM

El último estado posible para el circuito del *boost*, consiste en que el transistor se encuentre en no conducción y el diodo en estado DCM. Para este caso, tanto la corriente que circula por el diodo como la que circula por el transistor son nulas. Esto da lugar a que la corriente a la entrada del convertidor sea, necesariamente, nula. Este caso se puede visualizar en la Figura 12. Así pues, i_C valdrá lo mismo que i_R, ya que, como se ha explicado anteriormente, la corriente i_L sería nula. Por otro lado, para este estado el valor de v_L es nulo.

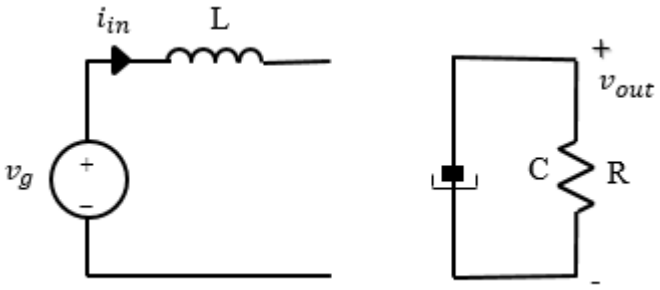

Figura 12. Esquema eléctrico del *boost* con mosfet off y diodo en DCM

En la Tabla 1 se puede observar un resumen de los resultados para i_C y para v_L dependiendo de los estados del circuito del convertidor conmutado, como se ha explicado anteriormente.

MOSFET = ON	**MOSFET = OFF**	
	DIODO = CCM	**DIODO = DCM**
$i_C = -i_R$	$i_C = (i_L - i_R)$	$i_C = -i_R$
$v_L = v_g$	$v_L = (v_g - v_{out})$	$v_L = 0$

Tabla 1. Resultados de i_C y de v_L dependiendo del estado del circuito

Utilizando los datos de la Tabla 1 y las ecuaciones 3.1.2 y 3.1.4, se pueden describir las ecuaciones que definen el modelo de planta como:

- Cuando el transistor esté conduciendo:

$$v_C(k) = v_C(\mathrm{k}-1) + [-i_R(\mathrm{k})] \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}} \tag{3.1.5}$$

$$i_L(k) = i_L(\mathrm{k}-1) + v_g(\mathrm{k}) \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}} \tag{3.1.6}$$

- Cuando el transistor no conduzca y el diodo esté en CCM:

$$v_C(k) = v_C(\mathrm{k}-1) + [i_L(k) - i_R(k)] \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}} \tag{3.1.7}$$

$$i_L(k) = i_L(\mathrm{k}-1) + [v_g(k) - v_o(k)] \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}} \tag{3.1.8}$$

- Cuando el transistor no conduzca y el diodo esté en DCM:

$$v_C(k) = v_C(\mathrm{k}-1) + [-i_R(k)] \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}} \tag{3.1.9}$$

$$i_L(k) = i_L(\mathrm{k}-1) + 0 \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}} = i_L(\mathrm{k}-1) \tag{3.1.10}$$

Además, en este caso como se aprecia en la Figura 7 e independientemente del estado del circuito, es decir, independientemente del estado de los conmutadores del circuito (diodo y transistor), se cumple:

$$v_C = v_{out} \tag{3.1.11}$$

$$i_L = i_{in} \tag{3.1.12}$$

Ahora que están definidas las ecuaciones que modelan el convertidor, se pueden emplear para elaborar los modelos en VHDL. Estas ecuaciones serán válidas tanto para el modelo *real* como para el modelo en coma fija.

Para definir las ecuaciones, procedemos a la elección de los elementos y las constantes que emplearemos para el diseño del convertidor elevador en VHDL. Para los elementos pasivos del circuito, se toma la inductancia de la bobina como 5 mH y para el condensador se toma un valor de 100 µF. Estos valores han sido tomados de [6], donde se explica el empleo de cada uno de los elementos anteriormente descrito. Para estos

valores, con una frecuencia de conmutación de 100 kHz (frecuencia a la que varía el estado del mosfet) y la lógica de control de 100 MHz (esta frecuencia es la que define el periodo de reloj), se obtiene una tensión nominal a la salida de 400 V, para una tensión de entrada eficaz nominal de 230 V a 50 Hz. Estos valores a emplear en los diseños de VHDL se detallan en la Tabla 2.

Bobina (L)	5 mH
Condensador (C)	100 µF
Periodo de reloj (dt)	10 ns
Periodo de conmutación	10 µs
Periodo del semiciclo de tensión de entrada	10 ms

Tabla 2. Resumen de elementos del circuito

3.1.1 Modelo *real* en VHDL para el *boost* sin pérdidas

Para la elaboración de este diseño se emplean las ecuaciones anteriores, desde (3.1.5) hasta (3.1.12). Este modelo es sencillo de implementar debido a que no es necesario un esfuerzo extra por parte del diseñador para conseguir una buena resolución, sino que el redondeo de los valores de las señales se realiza por software automáticamente con una precisión de un tipo coma flotante con 64 bits de resolución. Pero como se ha visto anteriormente, este tipo de datos no es verificable en FPGA, ya que no es sintetizable.

De todos modos, debido a la exactitud de los resultados de las ecuaciones con este tipo numérico, es crucial el correcto funcionamiento de este modelo para comparar, como se verá en el capítulo siguiente, este modelo *real* con el modelo en coma fija.

Este modelo tiene dos procesos, uno secuencial (llamado en el código *Asignacion*) y otro combinacional (conocido en el código como *switchMUX*), con los que se realiza el diseño completo del convertidor elevador antes nombrado y se pueden observar en Código 1 y Código 2. El modelo completo se puede ver en el Anexo (Modelo *Boost Real* Sin Pérdidas).

En estos códigos, se han simplificado los términos constantes $\frac{\Delta t}{C}$ e $\frac{\Delta t}{L}$ como dtC y dtL. Siendo estos últimos la fracción del incremento de tiempo definido en el código como dt y antes representado como Δt, por el valor del condensador y la bobina respectivamente. De esta forma se evita realizar una división en cada paso de integración.

```
Asignacion : process (Clk, Reset)
begin
	if Reset = '1' then
		Vo <= Vini;
		Ii <= Iini;
	elsif rising_edge(Clk) then
		Vo <= Vo + ic*dtC;
		Ii <= Ii + vl*dtL;
	end if;
end process Asignacion;
```

Código 1. Proceso secuencial del boost tipo *real* sin pérdidas

```
switchMUX : process (Mosfet, Vg, Ir, Ii, Vo)
begin
	if Mosfet = '1' then		-- mosfet = on -- corto cto
		vl <= Vg;
		ic <= -Ir;
	else				-- mosfet = off -- cto abto
		if Ii > 0.0 then	-- Diodo en CCM
			vl <= (Vg-Vo);
			ic <= (Ii-Ir);
		else			-- Diodo en DCM
			vl <= 0.0;	-- Ii=0
			ic <= -Ir;
		end if;
	end if;

end process switchMUX;
```

Código 2. Proceso combinacional del boost tipo *real* sin pérdidas

3.1.2 Modelo coma fija en VHDL para el *boost* sin pérdidas

La realización de este diseño es más compleja que la anterior, debido a que la elección de los tamaños de las señales es criterio del diseñador. Así pues, pese a que el código es muy parecido al del modelo *real*, el tamaño de las señales es fundamental para obtener unos resultados verídicos y no falseados debido a un redondeo erróneo por falta de resolución en los decimales o debido al desbordamiento (más conocido en inglés como *overflow*) por falta de resolución en los bits que definen la parte entera.

Para describir este diseño emplearemos una biblioteca de VHDL llamada *sfixed*, que permite la descripción de números en coma fija (también conocido como formato QX.Y como se ha comentado en el estado del arte) en VHDL. Esta definición permite tanto simular como sintetizar el modelo que se diseñe con este tipo de datos. Existen datos en coma fija con y sin signo. En este caso, se emplea un tipo de datos en coma fija con signo, debido a la posibilidad de corrientes o tensiones negativas, en sentido contrario al de definición, en el convertidor conmutado.

La notación QX.Y para coma fija consiste en emplear X bits para la parte entera, Y bits para la parte decimal y un bit de signo extra. Así el número total de bits empleado para un tipo Q2.3 sería 2 + 3 + 1 = 6 bits. Como ejemplo, el número “010101” representado en Q2.3, representa 010.101, es decir, 2,625. Otra forma de verlo es dado el número “010101” que se corresponde con 21 en notación binaria clásica y dividirlo

entre 2^3 (en notación general, dividirlo entre 2^Y), así su resultado sería $21/2^3 = 2{,}625$. Por otro lado, para el mismo número "010101" si por ejemplo se emplea la notación Q4.1, en este caso el valor numérico sería 01010.1 = 10,5. También es posible que X o Y sean negativas. Por ejemplo, para ese mismo número "010101", empleando la notación Q-2.7, cuyo valor sería 0.0010101 = 0,1640625. Visto de la otra forma $21/2^7 = 0{,}1640625$. Además se pueden representar números negativos. Para ello basta con poner un 1 en el primer bit, que se interpretará como muestra la Figura 13, es decir, en complemento a 2.

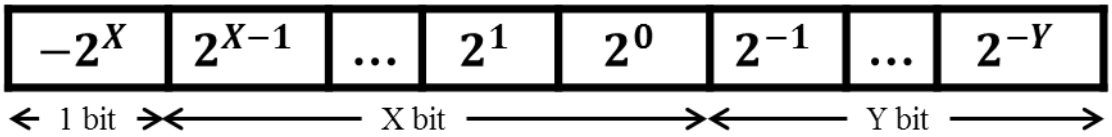

Figura 13. Formato de una señal QX.Y

Uno de los pasos más importantes consiste en la elección de tamaños para las constantes del circuito. En el caso del modelo real existen dos constantes dtL, con valor $2 \cdot 10^{-6}$, y dtC, de valor $0{,}0001$. Para poder representar estos valores en coma fija, se escogen los tamaños Q-18.35 y Q-13.30 respectivamente. Uno de los requisitos para la elección de estos tamaños es que no superen los 18 bits, ya que la FPGA que se empleará, *Xilinx Spartan* 3, tiene multiplicadores de 18 bits como se puede ver en la hoja de datos [12] que aceleran el proceso de multiplicación si se emplean señales de este tamaño o menores. Por lo tanto se debe localizar el bit de la posición más significativa para representar esos datos y otros 17 bits, como máximo, de posiciones menos significativas para conseguir una representación de las constantes sin perder velocidad de procesamiento (son 17 bits y no 18 los bits a añadir porque hay que tener en cuenta el bit de signo que añade el formato *sfixed*). En la Tabla 3 se puede observar el proceso para la elección de los tamaños para dtL y dtC.

	Valores	Mayor contiguo de base 2	Menor contiguo de base 2	Índice mayor	Formato en QX.Y
dtL	$2 \cdot 10^{-6}$	$2^{-18} = 3{,}8 \cdot 10^{-6}$	$2^{-19} = 1{,}9 \cdot 10^{-6}$	-18	Q-18.35
dtC	0,0001	$2^{-13} = 1{,}2 \cdot 10^{-4}$	$2^{-14} = 6{,}1 \cdot 10^{-5}$	-13	Q-13.30

Tabla 3. Resoluciones necesarias para las constantes del *boost* en coma fija

Además, este modelo de planta diseñado será verificado junto a un modelo de control definido en [3], por lo que los tamaños de las señales de entrada y de salida del circuito ya vienen definidos como se muestra en Código 3 y se corresponden con entradas de 13 bits con signo y salidas de 12 bits más signo ambas del tipo *std_logic_vetor* porque es lo estándar en VHDL debido a que *sfixed* no esta tan extendido. La definición del tipo *std_logic_vector* consiste en un vector de bits del tamaño definido entre paréntesis. Esta definición no tiene signo implícito, depende de cómo sea tratada por el usuario, y es un tipo propio de VHDL.

```
entity BoostQXYSP is
   port(
       --In
       Clk : in std_logic;
       Reset : in std_logic;
       Mosfet : in std_logic;                          -- On = '1', off = '0'
       Vg : in std_logic_vector(12 downto 0);          -- (Q9.3)
       Ir : in std_logic_vector(12 downto 0);          -- (Q3.9)
       -- Out
       Iin : out std_logic_vector(11 downto 0);        -- (Q3.9)
       Vout : out std_logic_vector(11 downto 0)        -- (Q10.2)
       );
end BoostQXYSP;
```

Código 3. Entradas y salidas del boost en coma fija

Para modelar los datos de tipo *std_logic_vector*, es necesario transformar estas señales a señales de tipo *sfixed*, gracias a las cuales se puede definir una parte decimal y otra parte entera, además del signo. En el código que se muestra en Código 4 y Código 5, estas señales correspondientes a las de entrada y salida del convertidor conmutado se definirán con el sufijo "_sf".

Al igual que en el modelo *real*, para el modelo en coma fija existen dos procesos que permiten definir el convertidor. Uno de los procesos es combinacional y otro secuencial que se pueden observar en los Código 4 y Código 5 respectivamente. En el proceso combinacional se emplean las señales de entrada y de salida en formato *sfixed* anteriormente descritas. Estas señales tienen resoluciones muy pequeñas comparadas con las que se emplean en el proceso secuencial, en el que la resolución del incremento de la tensión de salida y de la corriente de entrada son críticos para la evolución de las señales.

```
       signal ic: sfixed(3 downto -9) := (others=>'0');
       signal vl: sfixed(10 downto -3) := (others=>'0');
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

       switchMUX : process (Mosfet, Vg_sf, Ir_sf, Ii_sf, Vo_sf, vl, ic)
       begin
              if Mosfet = '1' then
                     vl <= resize(Vg_sf,  vl);
                     ic <= resize((-Ir_sf), ic);
              else
                     if Ii_sf > to_sfixed(0, Ii_sf) then
                            vl <= resize((Vg_sf - Vo_sf), vl);
                            ic <= resize((Ii_sf - Ir_sf), ic);
                     else
                            vl <= to_sfixed(0, vl);
                            ic <= resize((-Ir_sf), ic);
                     end if;
              end if;

       end process switchMUX;
```

Código 4. Proceso secuencial de un boost tipo coma fija sin pérdidas

Como se puede ver en Código 4, los tamaños escogidos inicialmente para las señales v_L e i_C se pueden derivar de los tamaños de las señales que lo definen, tal y como se ve

en la Tabla 4. Para el caso de v_L se escogen tres bits para definir los decimales, ya que, v_g tiene 3 bits de resolución para la parte decimal frente a los 2 bits que posee Vo_sf. De la misma manera se toman 10 bits para la parte entera aunque al realizar la resta serían 11 bits los necesarios para esta parte, se entiende que el resultado de esta resta no va a saturar, quedando definida v_L como Q10.3. Solo se toma en cuenta este caso ya que es el que más resolución necesita, ya que Vg_sf tiene menor número de bits. Siguiendo el mismo planteamiento que para la elección del tamaño de v_L, i_C queda definido como un Q3.9, esto se deriva de los tamaños de las señales Ii_sf e Ir_sf, que pese a que la resta de esas señales requiere un formato de Q4.9 la resta en este diseño nunca implica desbordamiento. Estas decisiones quedan detalladas en la Figura 14 y en la Figura 7.

```
	signal Ii_res: sfixed(3 downto -21) := (others=>'0');
	signal Vo_res : sfixed(10 downto -22) := (others=>'0');
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
	Asignacion : process (Clk, Reset)
	begin
		if Reset = '1' then
			Vo_res <= Vini;
			Ii_res <= Iini;
		elsif rising_edge(Clk) then
			Vo_res <= resize((Vo_res + Vo_inc21), Vo_res);
			Ii_res <= resize((Ii_res + Ii_inc21), Ii_res);
		end if;
	end process Asignacion;
```

Código 5. Proceso combinacional de un boost tipo coma fija sin pérdidas

La elección de tamaños de las señales del proceso secuencial aparece reflejada en la

Figura 14, pero a continuación se analizará cada una de las elecciones, ya que, las dimensiones de las señales se disparan en este proceso debido a la necesidad de tener suficiente resolución para la parte entera de las señales Vo_res e Ii_res, y además suficiente resolución en los decimales para que la variación en cada incremento sea lo más real posible.

Operation	Result Range
A + B	Max(A'left, B'left)+1 downto Min(A'right, B'right)
A - B	Max(A'left, B'left)+1 downto Min(A'right, B'right)
A * B	A'left + B'left+1 downto A'right + B'right

Tabla 4. Dimensiones para los resultados de las operaciones aritméticas en sfixed

Los productos dtC*ic y dtL*vl, se almacenan, según lo estipulado en la Tabla 4 y según muestra la

Figura 14, en Vo_inc de tamaño Q-9.39 y en Ii_inc de tamaño Q-7.38 respectivamente. Estos resultados tienen demasiados bits para ser almacenados y operar con ellos, que implicaría mayor tiempo de procesamiento y más recursos para una resolución ligeramente mejor, que como se verá en el capítulo de resultados, no es notable respecto a la resolución obtenida con menos bits que permiten el empleo de menos recursos y

tiempos menores. Además, gracias al modelo *real*, podemos saber los tamaños de los incrementos y de las señales para ajustar los tamaños de Vo_res y de Ii_res a los valores obtenidos anteriormente, y consecuentemente se redimensionarán Vo_inc e Ii_inc, ya que no necesitan más bits de los que se estimen para los decimales de Vo_res y de Ii_res.

Para la parte entera de Vo_res, el valor de la tensión de salida típico es 400 V. Este valor se puede representar con 9 bits, debido a que $2^8 = 256 < 400 < 2^9 = 512$, pero se toman 10 bits, que permiten representar hasta 1024, para dar un margen suficiente y evitar desbordamientos en transitorios grandes. Para la parte decimal, el incremento que se añade a la parte entera, Vo_inc, puede estar definido por un valor típico para i_C de 0,75, con lo que tenemos un valor para Vo_inc de $0{,}75 \cdot 10^{-4}$ (dtC·ic) así pues, se estiman 14 bits para la resolución decimal en este caso, porque $2^{-14} = 0{,}6 \cdot 10^{-4} < 0{,}75 \cdot 10^{-4} < 2^{-13} = 1{,}2 \cdot 10^{-4}$. Las líneas anteriores se pueden resumir con la ecuación:

$$\log_2\left(\frac{X}{\Delta X}\right) = \log_2\left(\frac{1024}{0{,}75 \cdot 10^{-4}}\right) \sim 24\ bits$$

En esta ecuación se especifica el número de bits necesario para representar tanto la parte entera como la parte decimal que en formato de coma fija se escribiría como Q10.14. Además, como se quieren representar incrementos distintos del típico, se toman N_v bits extra para su representación. En este caso, como se verá en el capítulo de resultado se añaden 8 bits extra. Así Vo_res queda definido como un Q10.22 y esto define consecuentemente la redimensión de Vo_inc en Vo_inc21 de formato Q-9.22. Por lo tanto se puede representar la resolución total de Vo_res como:

$$\log_2\left(\frac{X}{\Delta X}\right) = \log_2\left(\frac{1024}{0{,}75 \cdot 10^{-4}}\right) + N_v \sim 24 + N_v \quad bits$$

De la misma manera como se llega a la resolución necesaria de Vo_res, se sigue el mismo planteamiento para Ii_res. Así para la parte entera se estimarían suficientes 3 bits, ya que un valor típico es 2 que se encuentra entre $2^1 = 2$ y $2^2 = 4$, pero al igual que en el caso anterior para evitar desbordamientos incluso en el transitorio se toman 3 bits que permiten representar hasta 8. Por otro lado, para la parte decimal se pueden tomar como valores típicos $v_L = 100\,V$ y $dtL = 2 \cdot 10^{-6}$, quedando Ii_inc= $2 \cdot 10^{-4}$ (dtL*vl). Así se consideran suficientes 13 bits, ya que $2^{-13} = 1{,}2 \cdot 10^{-4}$ y $2^{-12} = 2{,}44 \cdot 10^{-4}$. Simplificando estas líneas se puede estimar la resolución como:

$$\log_2\left(\frac{X}{\Delta X}\right) = \log_2\left(\frac{8}{4 \cdot 10^{-4}}\right) \sim 16\ bits$$

Así se podría definir Ii_res como Q3.13, pero son necesarios N_i bits extra para tener resolución en el incremento y no que solo sea posible representar el valor típico

anterior. Así añadiendo los 8 bits de resolución extra como se verá en el capítulo de resultados, se queda Q3.21 como tamaño para Ii_res. Por lo tanto se puede definir la resolución necesaria para Ii_res con la siguiente fórmula:

$$\log_2\left(\frac{X}{\Delta X}\right) = \log_2\left(\frac{8}{4 \cdot 10^{-4}}\right) + N_i \sim 16 + N_i \ bits$$

Toda esta explicación se puede apreciar detalladamente en la

Figura 14, donde se representa el esquema del convertidor elevador en coma fija con los tamaños elegidos para cada una de las señales. Los tamaños que se determinarán en el capítulo de pruebas, ya que se realiza un análisis de resolución sobre ellos, están acompañados de *.

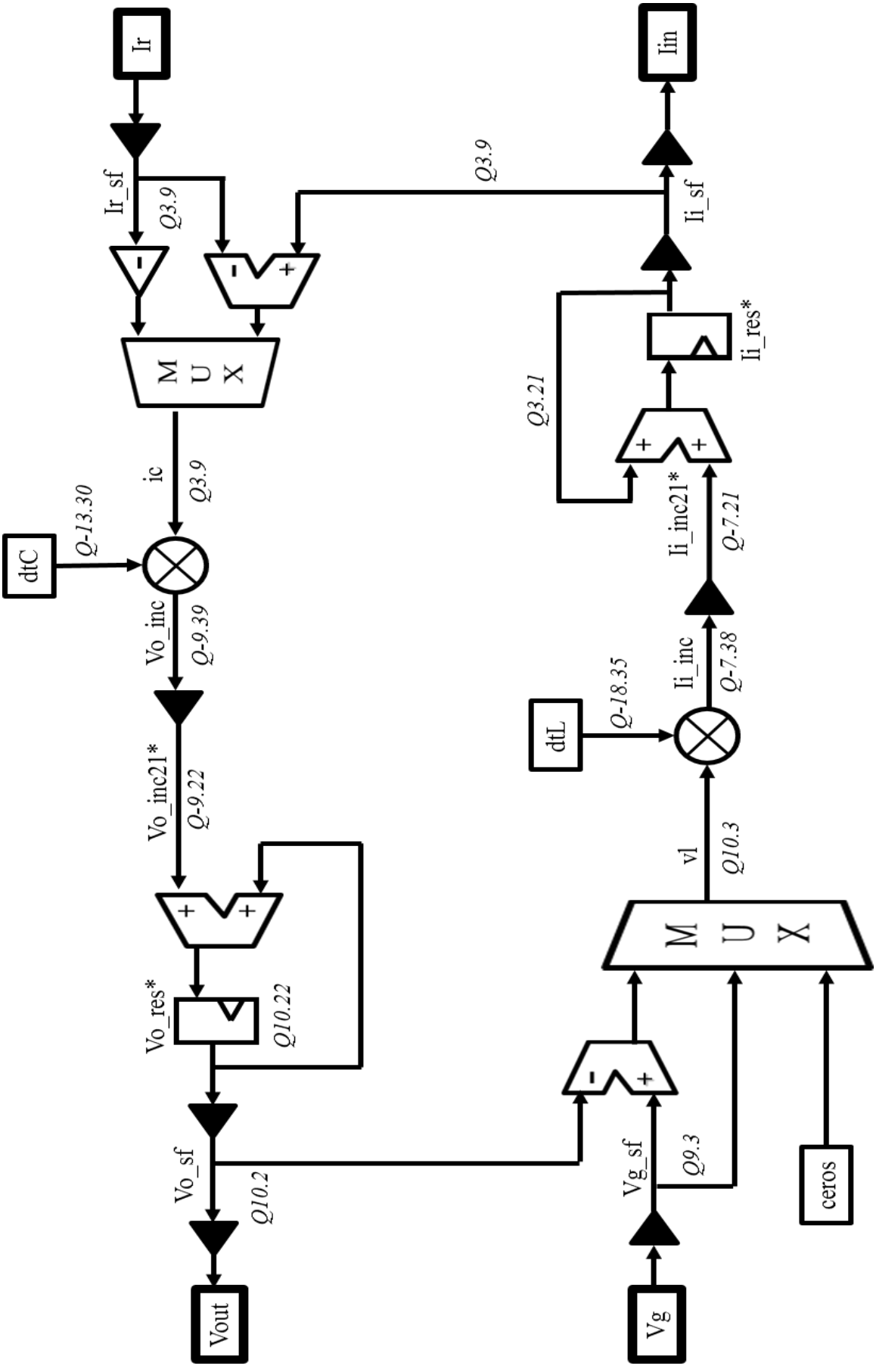

Figura 14. Esquema del circuito en coma fija

3.2. La influencia de las pérdidas

El diseño anterior representa los elementos ideales del circuito del convertidor conmutado, pero no se corresponde con el modelo real del convertidor analógico. Es decir, en todo circuito existen pérdidas, desde las relacionadas con el cable que une los elementos hasta las pérdidas propias de cada componente. Estas últimas pérdidas son, normalmente, mucho mayores que las correspondientes a las pérdidas del cable, que en el próximo diseño se tomarán como nulas.

La forma de onda para las señales de salida del modelo del convertidor conmutado sin pérdidas en lazo abierto, es decir, con un ciclo de trabajo contante en este caso de un 50% y con resistencia de salida fija, se pueden ver en la Figura 15. Lo que ocurre en esta figura es que la energía que llega al circuito se almacena y se transforma, no se disipa debido a que no hay elementos disipativos en este circuito como resistencias salvo la propia carga de salida, así que la bobina y el condensador del *boost* hacen de circuito resonante. Por lo tanto, se concluye que la solución del modelo sin pérdidas es poco amortiguada y tiene un periodo de estabilización muy grande.

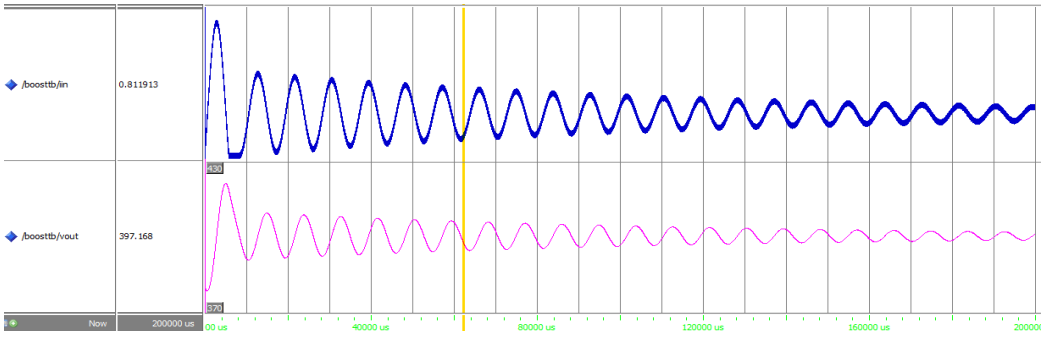

Figura 15. Tensión de salida y corriente de entrada para el *boost* sin pérdidas en *real*

Figura 16. Tensión de salida y corriente de entrada para el *boost* con pérdidas en *real*

Por otro lado, para el modelo con pérdidas, el resultado es menos amortiguado y el periodo de estabilización es menor como se puede ver en la Figura 16. En esta figura se representa nuevamente el modelo del lazo abierto con las mismas condiciones del caso anterior para un modelo que toma en cuenta las pérdidas. Pero en este modelo lo que ocurre es que la energía pasa de la bobina al condensador y se va disipando en los

elementos que representan las pérdidas, por lo que tarda muchos menos ciclos en estabilizarse ya que la energía no pasa íntegramente del condensador a la bobina como ocurría en el modelo sin pérdidas.

Para la simulación de estos modelos se ha realizado un *testbench* en el que se dan valores a las señales del boost y se añaden los periodos de reloj y de conmutación para poder ver la evolución del circuito. Lo que se persigue en este testbench, que simula el lazo abierto del boost, es ver el comportamiento del convertidor solo sin un control que varíe su salida y pueda tapar los posibles errores en el diseño. Con esto se persigue mostrar la importancia de las pérdidas en el modelo y la necesidad de tenerlas en cuenta en la realización de un modelo lo más parecido al funcionamiento real de la planta analógica.

3.3. Modelo de un convertidor elevador con pérdidas

Para modelar las pérdidas que sufre el convertidor elevador antes especificado es necesario conocer cada uno de sus componentes. Es decir, imitar el comportamiento real del circuito analógico en VHDL, y no el modelo ideal definido en el apartado 3.1. Para ello, en este diseño, se introducen las pérdidas conocidas como pérdidas de primer orden, que son las predominantes, aunque existan modelos más detallados para incluir otras pérdidas que se aproximan mejor al modelo real del convertidor analógico. Sin embargo, los modelos se complicarían bastante sin añadir apenas precisión.

El primer elemento que puede añadir pérdidas al circuito del convertidor elevador de la Figura 7 es la bobina. Las pérdidas que añade son en primera aproximación las pérdidas del cobre del inductor, que se pueden modelar con una resistencia en serie como muestra la Figura 17. Esta resistencia R_L se incorpora al circuito para modelar el *boost*.

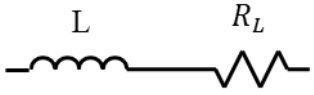

Figura 17. Modelado de las pérdidas de la bobina

El segundo elemento que añade pérdidas es el condensador. Este añade pérdidas que se pueden modelar en primera aproximación con una resistencia en serie. De hecho esta resistencia es tan común que se conoce como resistencia equivalente en serie, más conocida por sus siglas ESR (Equivalent Series Resistance). Con esta resistencia en serie con el condensador como se muestra en la Figura 18, se consigue modelar las pérdidas de potencia.

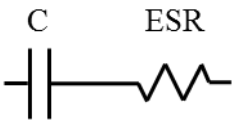

Figura 18. Modelado de las pérdidas del condensador con una resistencia ESR

El siguiente elemento a analizar es el transistor mosfet, que al igual que los anteriores, sus pérdidas se pueden asemejar a una resistencia en serie en primera aproximación. Así su circuito equivalente simplificado sería un interruptor ideal con una resistencia en serie como se muestra en la Figura 19.

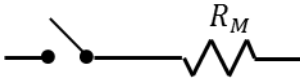

Figura 19. Circuito equivalente del transistor mosfet con pérdidas

El último elemento del circuito, que también añade pérdidas es el diodo. En este caso su circuito equivalente en primera aproximación se modela con una fuente de tensión a la que si se añade una resistencia en serie se pueden modelar pérdidas de segundo orden sin aumentar la complejidad del circuito excesivamente como se muestra en la Figura 20.

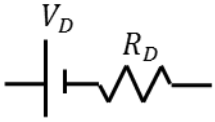

Figura 20. Circuito equivalente del diodo con pérdidas

Por lo tanto, la forma de modelar el circuito del convertidor con pérdidas es analizando su circuito equivalente, formado por los circuitos equivalentes de cada uno de sus componentes antes definidos. Este circuito final, que se puede ver en la Figura 21, será el circuito a analizar a lo largo de este punto y del que se desarrollarán los códigos VHDL tanto para el modelo con señales de tipo *real* como para el modelo con señales en coma fija.

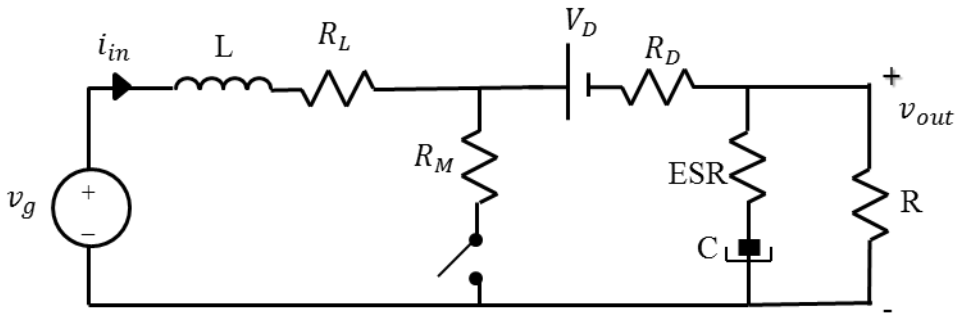

Figura 21. Circuito equivalente del *boost* con pérdidas

La manera de analizar este circuito es similar a la ya vista en el apartado 3.1 para el modelo sin pérdidas de la Figura 8, por lo tanto, se hará mayor hincapié en las diferencias entre ambos circuitos.

En primer lugar, las ecuaciones de estado tanto para este modelo, como para el modelo visto anteriormente son las mismas, siento v_L la tensión en la bobina e i_C la corriente que atraviesa el condensador:

$$i_L(k) = i_L(\mathrm{k}-1) + v_L(\mathrm{k}) \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}} \tag{3.4.1}$$

$$v_C(k) = v_C(\mathrm{k}-1) + i_C(\mathrm{k}-1) \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}} \tag{3.4.2}$$

La diferencia principal se da en el estado del circuito en función del estado de los conmutadores. Es decir, los estados en función de los conmutadores van a ser los tres estados vistos para el apartado anterior (*mosfet* en conducción, *mosfet* sin conducir y CCM y *mosfet* sin conducir y DCM), pero la disposición de los elementos para cada caso varía debido a la adición de nuevos elementos para modelar las pérdidas.

Para el primer caso en el que el mosfet se encuentra en conducción, el circuito equivalente se puede ver en la Figura 22. Para este caso hay que contemplar las resistencias R_L y R_M al hallar v_L, mientras que para el caso de i_C coincide con i_R independientemente del valor de ESR. Así la tensión que cae sobre la bobina, v_L es v_g menos la tensión que cae en cada una de las resistencias (R_L y R_M).

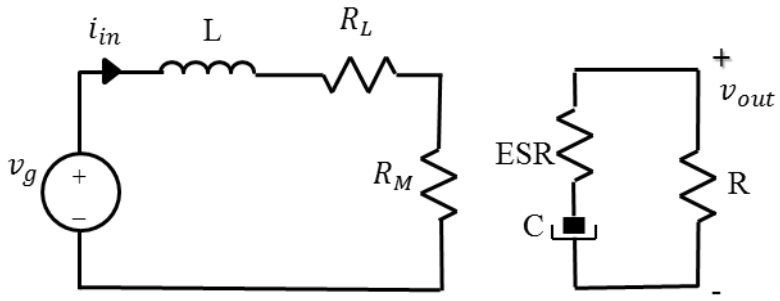

Figura 22. Esquema eléctrico del *boost* con pérdidas y mosfet on

Para el siguiente caso en el que el mosfet se encuentra en no conducción y el diodo está en CCM, el estado del circuito es el que muestra la Figura 23. En este caso la definición de v_L varía respecto al modelo sin pérdidas. Así v_L es igual a la tensión de entrada v_g menos la tensión de salida v_{out}, menos la tensión del diodo equivalente V_D, y menos la tensión que cae en las resistencias de pérdidas involucradas en el lazo que se

está analizando (R_L y R_D). Por otro lado la corriente i_C en este caso es la diferencia entre la corriente de entrada y la de salida, i_L-i_R, igual que en sin pérdidas

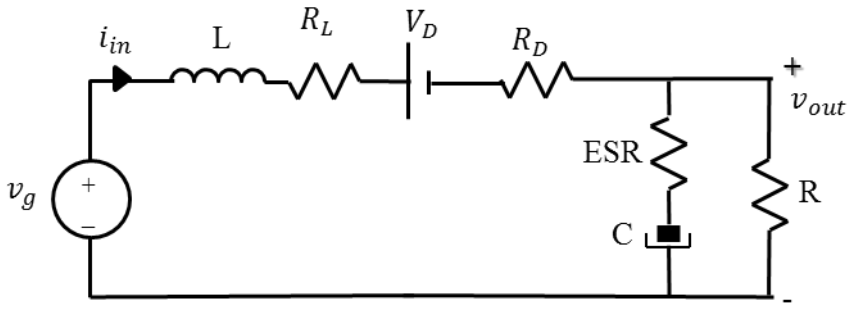

Figura 23. Esquema eléctrico del *boost* con pérdidas, mosfet off y diodo en CCM

El último caso en el que el mosfet está en no conducción y el diodo en estado DCM. El resultado es igual al modelo sin pérdidas y se puede ver en la Figura 24. En este caso la tensión v_L es nula, mientras que la corriente del condensador i_C es igual a la de salida i_R.

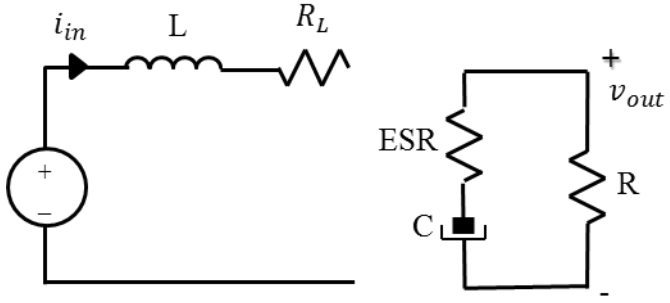

Figura 24. Esquema eléctrico del *boost* con pérdidas, mosfet off y diodo en DCM

Como ya se ha explicado anteriormente la obtención de las variables i_c y v_L, la Tabla 5 muestra un resumen de estos resultados para las variables de estado dependiendo del estado del circuito en cada caso.

MOSFET = ON	**MOSFET = OFF**	
	DIODO = CCM	**DIODO = DCM**
$i_C = -i_R$	$i_C = i_L - i_R$	$i_C = -i_R$
$v_L = v_g - i_L \cdot (R_L + R_M)$	$v_L = v_g - v_{out} - V_D - i_L \cdot (R_L + R_D)$	$v_L = 0$

Tabla 5. Resultados de i_C y de v_L dependiendo del estado del circuito con pérdidas

Recopilando lo obtenido anteriormente, se pueden definir las ecuaciones que modelan el convertidor conmutado con pérdidas dependiendo del estado en el que se encuentre el

circuito uniendo los resultados plasmados en la Tabla 5 junto a las ecuaciones (3.4.1) y (3.4.2):

- Cuando el transistor esté conduciendo:

$$v_C(k) = v_C(\mathrm{k}-1) + [-i_R(\mathrm{k})] \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}} \tag{3.4.3}$$

$$i_L(k) = i_L(\mathrm{k}-1) + [v_g(\mathrm{k}) - i_L(\mathrm{k}-1) \cdot (R_L + R_M)] \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}} \tag{3.4.4}$$

- Cuando el transistor no conduzca y el diodo esté en CCM:

$$v_C(k) = v_C(\mathrm{k}-1) + [i_L(k) - i_R(k)] \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}} \tag{3.4.5}$$

$$i_L(k) = i_L(\mathrm{k}-1) + [v_g(k) - v_{out}(k) - V_D - i_L \cdot (R_L + R_D)] \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}} \tag{3.4.6}$$

- Cuando el transistor no conduzca y el diodo esté en DCM:

$$v_C(k) = v_C(\mathrm{k}-1) + (-i_R(k)) \cdot \frac{\Delta \mathrm{t}}{\mathrm{C}} \tag{3.4.7}$$

$$i_{in}(k) = i_L(\mathrm{k}-1) + 0 \cdot \frac{\Delta \mathrm{t}}{\mathrm{L}} = i_L(\mathrm{k}-1) \tag{3.4.8}$$

Por otro lado, la ESR influye en la tensión de salida, por lo que v_{out} ya no coincide con la tensión que cae en el condensador, v_C. Con esto se determina que las siguientes ecuaciones son válidas para definir el modelo con pérdidas independientemente del estado de conmutación del circuito.

$$v_C = v_{out} - i_C \cdot ESR \tag{3.4.9}$$

$$i_L = i_{in} \tag{3.4.10}$$

Una vez que tenemos las ecuaciones que describen el circuito del convertidor conmutado con pérdidas, se pueden emplear para realizar los modelos en VHDL tanto para señales descritas en *real* como para señales definidas en coma fija.

Para terminar de definir el convertidor conmutado es necesario obtener los valores de los elementos del sistema que serán los mismos que los escogidos anteriormente para el modelo sin pérdidas, ya que así se podrán comparar ambos casos entre sí. Pero hay elementos que no tienen un valor definido previamente ya que no se encontraban presentes en el modelo sin pérdidas. Estos elementos que representan las pérdidas tienen valores que han sido sacados de las hojas de datos de los componentes que se imitan en este modelo en VHDL, todo esto se recopila en la Tabla 6, junto a los datos anteriores útiles para este caso con pérdidas. Se toma un valor nulo de R_D ya que estas pérdidas son consideradas secundarias y no añade variaciones críticas al diseño con pérdidas.

Bobina (L)	5 mH
Condensador (C)	100 µF
ESR	3,316 Ohm
R_L	0,06965 Ohm
R_M	0,18 Ohm
R_D	0 Ohm
V_D	1,45 V

Tabla 6. Resumen de elementos del circuito con pérdidas

3.3.1 Modelo *real* en VHDL para el *boost* con pérdidas

Este modelo se lleva a cabo gracias a las ecuaciones descritas en el apartado anterior, tanto la definición de las ecuaciones de la tabla 5 como las ecuaciones de la (3.4.7) a la (3.4.10). Para ello, igual que en el caso sin pérdidas, existen dos procesos, uno secuencial (llamado en el código *Asignacion*) y otro combinacional (conocido en el código como *switchMUX*), con los que se realiza el diseño completo del convertidor elevador con pérdidas antes nombrado y se pueden observar en Código 6 y Código 7. El modelo completo se puede ver en el Anexo 1 (Modelo del *Boost* Real Con Pérdidas).

```
Asignacion : process (Clk, Reset)
begin
	if Reset = '1' then
		Vc <= Vini;
		Ii <= Iini;
		Vo <= Vini;
	elsif rising_edge(Clk) then
		Vc <= Vc + ic*dtC;

		Ii <= Ii + vl*dtL;
		Vo <= Vc + ic*ESR;
	end if;
end process Asignacion;
```

Código 6. Proceso secuencial del boost tipo *real* con pérdidas

```
switchMUX : process (Mosfet, Vg, Ir, Ii, Vo)
begin
	if Mosfet = '1' then		-- mosfet = on -- corto cto
		vl <= Vg-Ii*(Rl+Rm);
		ic <= -Ir;
	else				-- mosfet = off -- cto abto
		if Ii > 0.0 then	-- Diodo en CCM
			vl <= Vg-Ii*(Rl+Rd)-Vd-Vo;
			ic <= Ii-Ir;
		else			-- Diodo en DCM
			vl <= 0.0;
			ic <= -Ir;
		end if;
	end if;
end process switchMUX;
```

Código 7. Proceso combinacional del boost tipo *real* con pérdidas

Los valores dtC y dtL definen las mismas constantes que para el caso sin pérdidas, $\frac{\Delta t}{C}$ e $\frac{\Delta t}{L}$. Las nuevas constantes ESR, Rl, Rd, Vd y Rm, tienen el valor que se ha definido en la Tabla 6. Como se puede ver en este caso, es necesario definir tanto la variable de estado Vc, como la tensión de salida Vo. Ya que a diferencia del modelo sin pérdidas en este caso estas variables no coinciden, además de definir ambas es necesario inicializarlas en el proceso secuencial.

3.3.2 Modelo coma fija en VHDL para el *boost* con pérdidas

Para este caso, como se ha explicado para el diseño sin pérdidas, los tamaños de las señales son claves para su correcto funcionamiento, y estos tamaños son elección del diseñador. Los elementos del circuito ya existentes para el modelo sin pérdidas mantienen su tamaño para este caso, exceptuando las señales que dependan de las nuevas pérdidas, aunque teniendo en cuenta que las entradas y salidas de este modelo serán las mismas que las del modelo sin pérdidas, finalmente no habrá gran diferencia entre las dimensiones de las señales de ambos modelos (con y sin pérdidas). Por lo tanto, hay que decidir los tamaños para las nuevas variables, es decir, para los valores de los elementos del circuito equivalente para el modelo con pérdidas de la Figura 21.

Para esto se verá en el capítulo siguiente cómo afecta la resolución de estas señales al resultado final del circuito. Así en la Tabla 7 se pueden ver unos tamaños típicos para las pérdidas de este, es decir, para los elementos de los circuitos equivalentes que sustituyen a los elementos del circuito, aunque la elección final está detallada en el siguiente capítulo.

Componente	Valor	Formato	Resolución
ESR	3,316 Ohm	Q2.6	8 bits
R_L	0,06965 Ohm	Q0.8	8 bits
R_M	0,18 Ohm	Q0.8	8 bits
R_D	0 Ohm	Q2.4	6 bits
V_D	1,45 V	Q2.6	8 bits

Tabla 7. Tabla con los valores y resoluciones típicas de las pérdidas

En los siguientes Código 7 y Código 8, se muestran los procesos secuencial y combinacional que definen el modelo del *boost* con pérdidas. El proceso secuencial es similar al del modelo *real*, con la diferencia de que este modelo debe añadir unos tamaños específicos a las señales siguiendo las indicaciones que se mostraban en la Tabla 4. Es necesario redimensionar varias señales, sobre todo las señales Vo_res, Vc_res e Ii_res, que sufren una realimentación en cada iteración. Por lo tanto es necesario usar la función *resize* de la biblioteca *sfixed*, que permite redimensionar una operación de suma o de resta, pero no permite hacer lo propio con las multiplicaciones. Por lo tanto se requiere el uso de variables auxiliares, como ya ocurría en el caso de los incrementos para el modelo sin pérdidas y en este caso también ocurre para la definición de ic y de vl, donde se emplean las variables aux1 y aux2. Estas variables varían su tamaño en función del tamaño necesario para abarcar las operaciones que almacenan, a diferencia de las señales anteriormente nombradas que mantienen sus tamaños del modelo sin pérdidas (vl, ic, Vo_res, Vc_res e Ii_res). El tamaño final acepado para estas variables se puede observar en la Figura 25, al igual que los tamaños del resto de señales aunque el estudio detallado está en el siguiente capítulo.

Para los tamaños de las señales de salida, Vo_res e Ii_res, se mantienen los tamaños del caso sin pérdidas, que comprendía los bits necesarios para representar un incremento más una resolución de 8 bits extra como quedará demostrado en el capítulo siguiente, en el cual, también se verificará que es resolución suficiente tanto para el modelo con pérdidas como para el modelo sin pérdidas. Los formatos de estas señales y del resto de señales empleadas se pueden ver en la Figura 25. En esta figura se acompañan de * las señales que van a ser analizadas en el capítulo de resultados y cuyos tamaños finales determinados tras la realización de varias pruebas son los que muestra esta figura.

```
signal Ii_inc : sfixed(-7 downto -42) := (others => '0');
signal Vc_inc : sfixed(-9 downto -39) := (others => '0');
signal ESR_ic : sfixed(6 downto -15) := (others => '0');

signal Ii_inc21 : sfixed(-7 downto -21) := (others => '0');
signal Vc_inc21 : sfixed(-9 downto -22) := (others => '0');

signal Ii_res: sfixed(3 downto -21) := (others=>'0');
signal Vc_res : sfixed(10 downto -22) := (others=>'0');
signal Vo_res : sfixed(10 downto -22) := (others=>'0');
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Ii_inc <= vl*dtL;
Ii_inc21 <= resize(Ii_inc, Ii_inc21);

Vc_inc <= ic*dtC;
Vc_inc21 <= resize(Vc_inc, Vc_inc21);

ESR_ic <= ic*ESR;

Asignacion : process (Clk, Reset)
begin
       if Reset = '1' then
              Vo_res <= Vini;
              Vc_res <= Vini;
              Ii_res <= Iini;
       elsif rising_edge(Clk) then
              Vc_res <= resize((Vc_res + Vc_inc21), Vc_res);
              Ii_res <= resize((Ii_res + Ii_inc21), Ii_res);

              Vo_res <= resize((Vc_res+ESR_ic),Vo_res);
       end if;
end process Asignacion;
```

Código 7. Proceso secuencial del boost en coma fija con pérdidas

```
signal aux1: sfixed(5 downto -17);
signal aux2 : sfixed(7 downto -17);

signal ic: sfixed(3 downto -9) := (others=>'0'); --  13 bits    (ic)
signal vl: sfixed(10 downto -7) := (others=>'0');
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

aux1 <= Ii_sf*(Rl+Rm);
aux2 <= Ii_sf*(Rl+Rd);

switchMUX : process (Mosfet, Vg_sf, Ir_sf, Ii_sf, Vo_sf)
begin
       if Mosfet = '1' then
              vl <= resize((Vg_sf - aux1),  vl);
              ic <= resize((-Ir_sf), ic);
       else
              if Ii_sf > to_sfixed(0, Ii_sf) then
                     vl <= resize((Vg_sf - aux2 - Vo_sf - Vd), vl);
                     ic <= resize((Ii_sf - Ir_sf), ic);
              else
                     vl <= to_sfixed(0, vl);
                     ic <= resize((-Ir_sf), ic);
              end if;
       end if;

end process switchMUX;
```

Código 8. Proceso combinacional del boost en coma fija con pérdidas

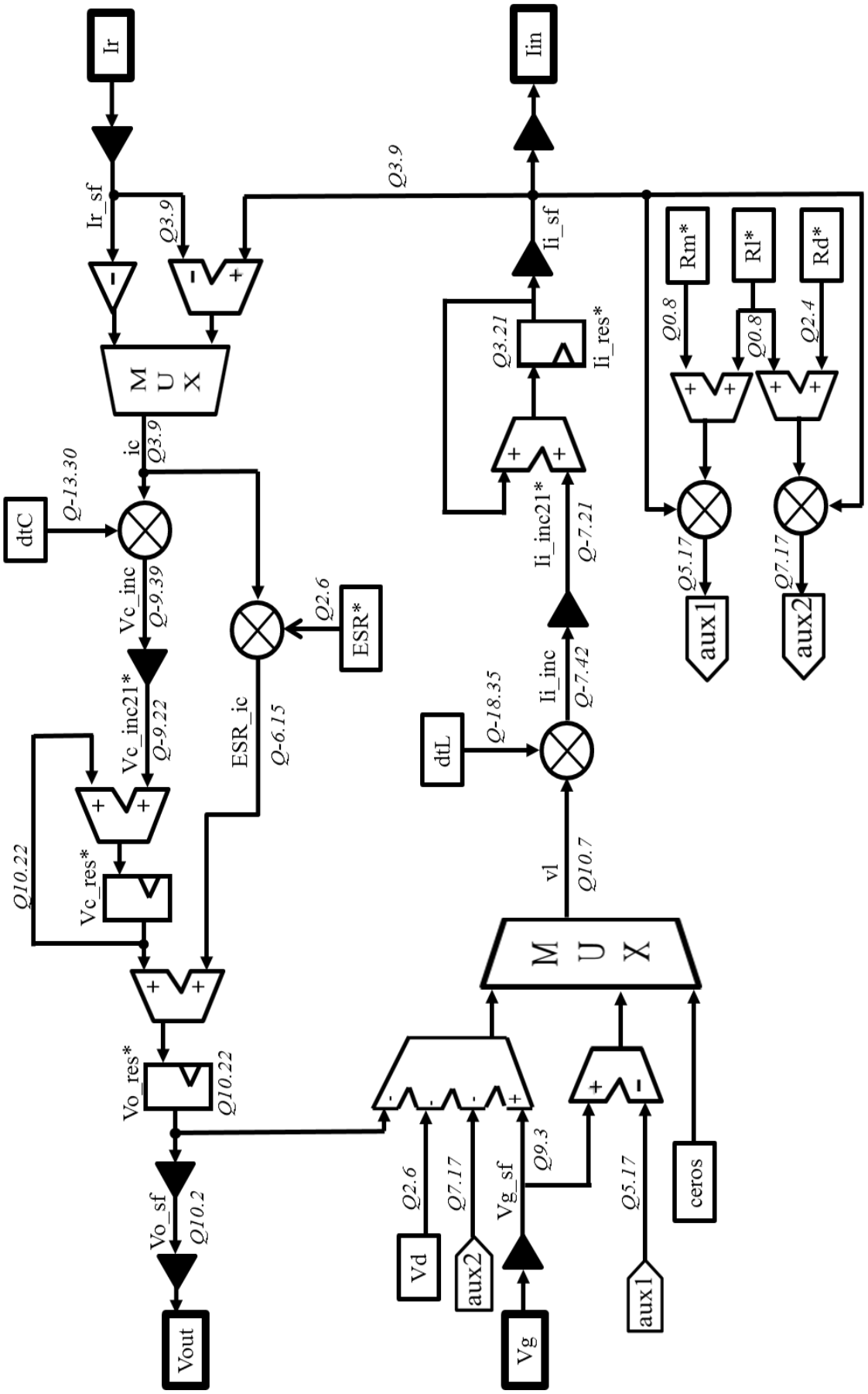

Figura 25. Esquema del *boost* con pérdidas en coma fija

3.4. Verificación del modelo en coma fija

Para llevar el modelo en coma fija a su verificación en FPGA es necesario realizar una serie de pasos previos pese a que el diseño del *boost* ya esté realizado. Estos pasos se llevarán a cabo mediante el programa Xilinx, concretamente la versión ISE Design Suite 14.4. Además la FPGA que se utilizará es la Spartan3 XC3S1000 que se programará gracias a un cable JTAG (del inglés *Joint Test Action Group*).

En primer lugar, para realizar una verificación hardware es necesario ajustar el periodo de reloj requerido por el modelo, ya que en caso contrario, tendrá lugar un error en las restricciones de tiempo (más conocidas por su nombre en inglés *timing constraint*). El periodo por defecto, debido a que es el periodo del reloj de la FPGA es de 20 ns, que no es suficiente para realizar las operaciones necesarias por ciclo de reloj del diseño del *boost* como se ve en la Figura 26, que requiere un periodo de trabajo de 37,809 ns. Por lo tanto se varía este reloj hasta conseguir un periodo menos restrictivo que permita el correcto funcionamiento del circuito.

	Met	Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
1	No	TS_Clk = PERIOD TIMEGRP "Clk" 20 ns HIGH 50%	SETUP HOLD	-17.809ns 1.597ns	37.809ns	183 0	2923443 0

Figura 26. Restricciones de tiempo de la verificación del boost sin pérdidas

Para el modelo del convertidor sin pérdidas solo, el periodo de reloj que permite cumplir las restricciones de tiempo para 8 bits de resolución, como se ha visto para el diseño en coma fija, es de 36 ns. Con esta variación se puede verificar el modelo del convertidor conmutado sin pérdidas y así obtener el área empleada y la frecuencia a la que trabaja.

Siguiendo el mismo procedimiento que en el modelo sin pérdidas, tras varias pruebas variando el periodo de reloj, se determina que para el modelo con pérdidas de 8 bits de resolución, el periodo necesario del *boost* es de 55 ns.

Para pasar estos diseños a la FPGA, como se ha visto que el periodo de reloj de la FPGA que aparece en [12] es de 20 ns y el periodo necesario por los modelos anteriores es mayor, se debe añadir un divisor de frecuencia.

El divisor de frecuencia se añade en el proyecto de Xilinx como un elemento más del Top, es decir como un componente que forma el conjunto a sintetizar. Lo que hace este divisor es recibir el periodo de reloj de la FPGA, 20 ns, y lo ralentiza hasta el periodo deseado. En este caso, lo que se ha hecho es triplicar este periodo para que sea suficiente tanto para el modelo sin pérdidas como para el modelo con pérdidas. Ya que con 60 ns se cumplen las restricciones de tiempo para ambos modelos.

El siguiente paso para llevar un diseño a su verificación en FPGA es crear un *testbench* que manipule las señales que no vayan a ser añadidas a los pines de entrada y salida de la FPGA. Entre estas señales se encuentra la señal mosfet que en este caso se le asigna un ciclo de trabajo constante. Mientras la señal de *reset* se puede añadir a una de las entradas de la FPGA, al igual que el reloj que se sincroniza con el reloj de la FPGA mediante un divisor de frecuencia incluido en la interfaz de Xilinx. Otras de las señales que se añaden en el *testbench* son la tensión de entrada y la corriente de salida.

Los valores de las señales del *testbench* se eligen para que tenga un comportamiento idéntico al ya realizado para simulación en lazo abierto. Este código se puede ver en el anexo *testbench* del boost en lazo abierto para FPGA. También se puede encontrar en este anexo el *testbench* empleado para el lazo cerrado con el nombre "*testbench* para el lazo cerrado para FPGA".

Por último, ya se podría introducir el modelo en FPGA, pero en este caso las comprobaciones y las medidas se van a realizar con un programa llamado ChipScope, de Xilinx, ya presentado en el estado del arte. Básicamente ChipScope funciona como un osciloscopio que permite medir los pines de la FPGA y que en este caso se empleará para ver las formas de onda. Para el empleo de este programa basta con añadir un fichero de definición y conexión de archivo de ChipScope en el que se puede seleccionar el número y tipo de restricciones para los disparos y las señales que se mostrarán.

Tras estos pasos, se puede realizar la verificación del convertidor tanto en lazo abierto como en lazo cerrado sobre FPGA y tomar los resultados gracias a ChipScope que permite ver las emulaciones hardware en un ordenador con este entorno.

Para tomar los datos con ChipScope de la FPGA se crean buses de datos con las señales a visualizar. Si tras esto se toman los datos requeridos, en este caso tensión de salida y corriente a la entrada, ocurre lo que muestra la Figura 27. Como se puede observar esta imagen no aporta ninguna información útil.

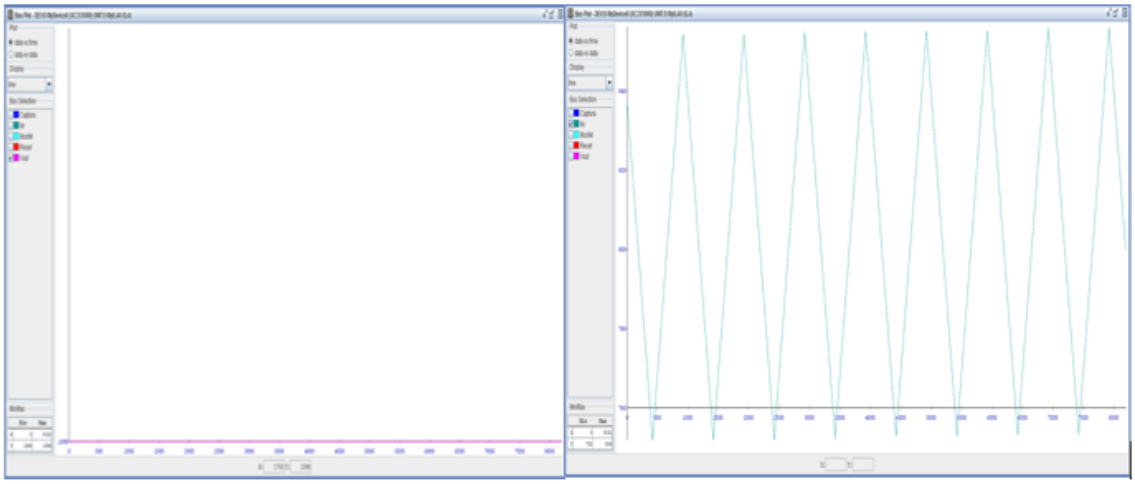

Figura 27. Emulación con ChipScope del boost en lazo abierto sin pérdidas, sin condiciones de captura. Con Vout a la izquierda e Iin a la derecha

La falta de información de esta señal se debe a dos cuestiones. En primer lugar, el boost ya está funcionando cuando se hace la captura con ChipScope, por lo que es muy posible que ya se encuentre en un estado de equilibrio. Lo que interesa en este caso es tomar los valores en el transitorio de la señal. Para conseguir esto, se crea una condición con la que se especifique que se toman los datos tras el flanco de bajada del reset. Así siempre será necesario pulsar el botón de reset para tomar datos en ChipScope pero esto garantiza que los valores mostrados son los valores del transitorio.

La segunda cuestión que influye en los datos tomados es debida al límite de almacenamiento de ChipScope. Para las señales que se quieren mostrar es necesario almacenar más de 20 bits, con este valor el número de bits máximo que se puede tomar para cada señal se reduce a 8192 bits máximo. Como el periodo de reloj es de 10 ns, se concluye que solo se pueden tomar bits hasta el instante de tiempo 81,92 us. Esto es muy breve y no permite ver la evolución de las señales. Por lo tanto se crea una segunda condición de captura para que se tome un dato de cada 1000. Con esto se logra representar 81,92 ms, tiempo suficiente para ver el transitorio. Este cambio provoca que la señal tomada en ChipScope se corresponda con la envolvente de la señal real como se muestra en las Figura 28 y Figura 29, en estas figuras se muestran las señales Vo e Ii obtenidas por simulación y por emulación del boost en lazo abierto para 8 bits de resolución.

La diferencia entre las señales obtenidas por simulación y por emulación se puede apreciar en el grosor de cada una. Esto se debe a que en emulación se toma un dato de cada 1000 por lo que se representa la envolvente de la señal. Para entender mejor este concepto se muestra la Figura 30 en la que se puede observar una ampliación de un intervalo de Vout. En esta imagen los puntos rojos representan los puntos que se capturan en emulación, uno de cada 1000 y el resto de puntos se descartan de la representación, ya que la memoria de almacenamiento para los valores capturados es muy limitada.

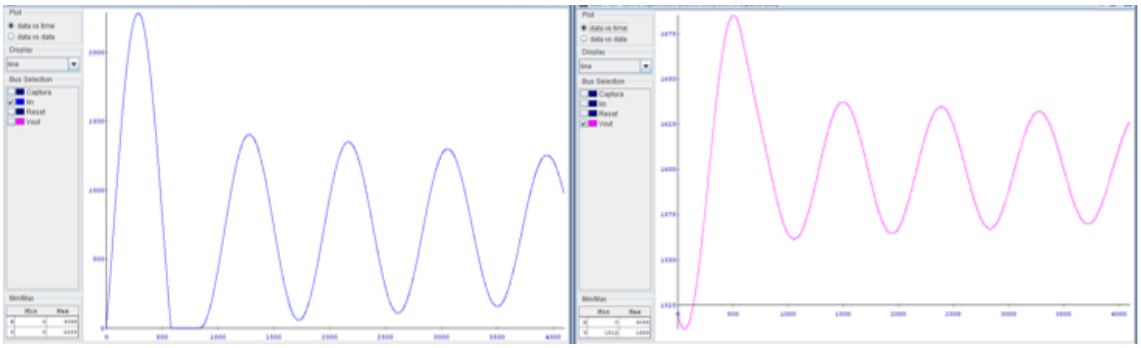

Figura 28. Emulación del lazo abierto del modelo sin pérdidas de Ii y de Vo

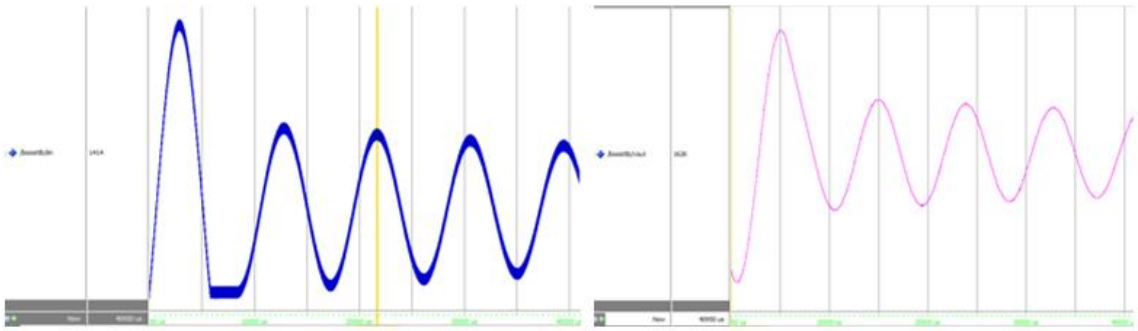

Figura 29. Simulación del lazo abierto del modelo sin pérdidas de Ii y de Vo

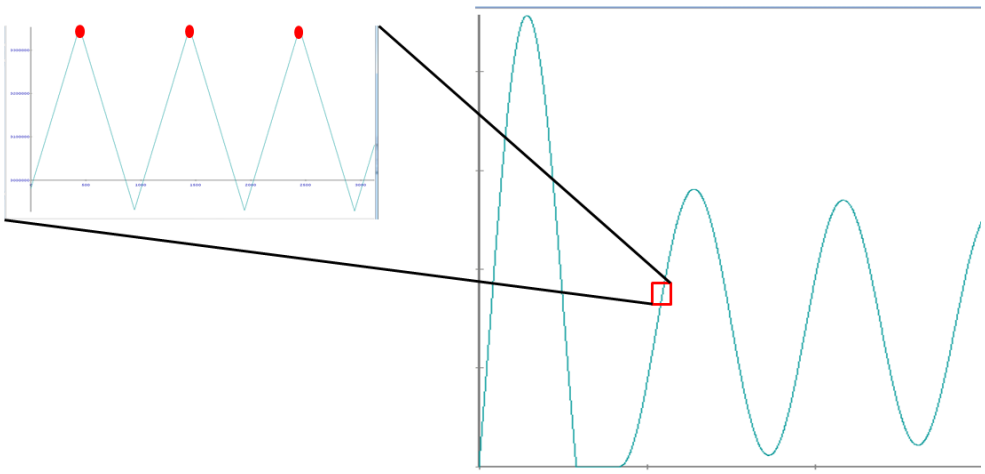

Figura 30. Emulación de vout y ampliación de un intervalo

4 RESULTADOS

En este capítulo se van a exponer los resultados finales que se perseguían en este trabajo. Además se mostrarán las pruebas realizadas hasta conseguir estas soluciones. Los objetivos anteriormente explicados se basan principalmente en la necesidad de reducir el tiempo de verificación de una fuente conmutada completa, es decir, del modelo de planta y del control conjuntamente. Los resultados que se muestran en este capítulo son en base a los modelos de planta explicados en el capítulo anterior.

Los resultados que se buscan se pueden clasificar en tres tipos. En primer lugar, se realizan varias pruebas para verificar la **precisión** y similitud del modelo emulable, el modelo de coma fija, con el modelo *real*. En segundo lugar, tras encontrar varias resoluciones que se adaptan al modelo real, se observan los **recursos**, tanto de área ocupada como la frecuencia de reloj necesaria, que consumen cada una de estas simulaciones en función del número de bits que se requiere. Con estos dos análisis anteriores se llevan a cabo los diseños mostrados en el capítulo anterior en las mejores condiciones para verificarlos en lazo cerrado, es decir junto al control de la planta. Los últimos resultados, objetivo de este trabajo consiste en los **tiempos** de simulación y de emulación que se requiere para el modelo real y para el modelo en coma fija. Así se comprobará cómo se reducen los tiempos de simulación y verificación para este modelo respecto al modelo mixto con un modelo de planta analógico junto con un control digital.

Por otro lado, como se ha explicado anteriormente, los primeros resultados se realizan en lazo abierto, para lo que se diseña un entorno de pruebas (más conocido en inglés como *testbench*) enfocado a los modelos del *boost* previamente diseñados y antes explicados. Así se comprueba la validez real de los modelos, ya que en lazo cerrado el control lleva la tensión de salida a un valor de consigna aunque el modelo sea erróneo. Una vez que estos modelos son testeados se comprueba su validez para el modelo en lazo cerrado junto al control diseñado en [3] y explicado anteriormente en el estado del arte.

La necesidad de realizar las pruebas en lazo abierto se observa claramente con la diferencia entre el modelo con pérdidas y sin pérdidas. En el punto 3.4, se veía cómo diferían las señales para el modelo con pérdidas y para el modelo sin pérdidas en lazo abierto. Pues bien, si esta simulación se realiza el lazo cerrado, ambas señales son muy parecidas, como se muestra en las Figura 31 y Figura 32, frente a lo visto para las Figura 15 y Figura 16.

Por este motivo, las pruebas se realizan en lazo abierto previamente para tener una imagen real del comportamiento del convertidor conmutado. Para posteriormente con

estos resultados, comprobar las mejoras obtenidas en lazo cerrado, es decir con el modelo completo de la fuente conmutada.

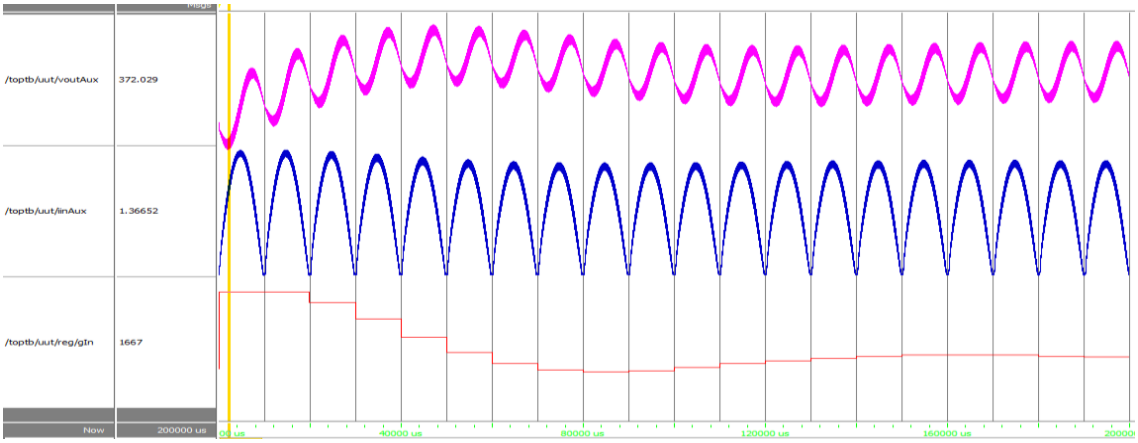

Figura 31. Simulación en lazo cerrado del modelo *real* con pérdidas

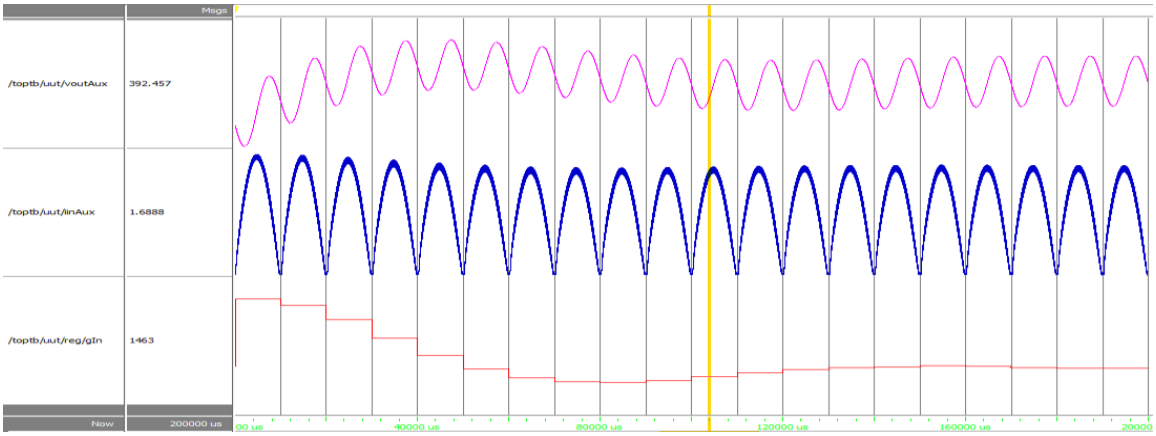

Figura 32. Simulación en lazo cerrado del modelo *real* sin pérdidas

4.1. Precisión

El concepto de precisión aparece en el diseño en coma fija, debido a que es necesario que el número de bits sea suficiente y esté en una escala adecuada para la representación del número oportuno. Por lo que a continuación distinguiremos entre los dos fallos principales en resolución, es decir, falta de resolución en la parte entera, y posteriormente falta de resolución en la parte decimal, que dependiendo de la exactitud en esta parte será más o menos trascendental.

La parte más relevante de los datos, los bits más significativos, siempre deben ser representables y no se debe dar lugar a casos de desbordamiento de datos, *overflow* en inglés. Si este fenómeno tiene lugar, el dato satura al mayor representable, como ejemplo, si se intenta representar el número 9, en binario “1001” en un formato Q2.3, nos dejaría los siguientes espacios: _ _ _._ _ _ . Como se ha visto en la Figura 13 la

representación de cada espacio sería $-2^2 + 2^1 + 2^0 + 2^{-1} + 2^2 + 2^{-3}$, con lo que la máxima aproximación a la que se puede llegar es: 011.111 que se corresponde con el número real 3.875, que dista mucho del valor deseado. Por ello siempre se debe tener suficiente resolución para la parte entera de un número representado en coma fija.

Por otra parte, la resolución de la parte decimal puede ser menos importante a la hora de representar un dato, a menos que como en este caso, esa resolución sea necesaria ya que importa el incremento con el que crece el dato. Así si se representa el número 3,316, que en coma fija se puede ver como 11.0101000011100101011, en un formato Q2.4, quedaría el número real 3,3125, que no difiere mucho del esperado, pero si en su lugar se representa ese dato con un formato Q2.2, el número real resultante sería 3,25, algo más alejado del valor a representar.

Cabe destacar que los fallos de desbordamiento son más graves, pero son mucho más fáciles de detectar, ya que aparecerían datos muy alejados de su valor ideal. Pero también es más fácil calcular su resolución, es decir, los bits a utilizar para la parte entera, ya que, casi siempre se conocen los valores mayores que deberían aparecer en una señal, y más así en nuestro caso en el que se ha realizado un diseño con aritmética *real* previo que ha permitido conocer los máximos de las señales de salida.

Tras este breve recordatorio, nos centramos en las resoluciones propias para el diseño del convertidor conmutado.

4.1.1 Resolución para el modelo sin pérdidas

El diseño del modelo sin pérdidas anteriormente explicado en coma fija, tiene su dificultad en la resolución apropiada para la tensión de salida y para la corriente de entrada ya que evolucionan en cada ciclo de reloj y este paso debe ser lo más similar al paso real del modelo de planta para que puedan alcanzar su valor de consigna.

Por lo tanto las señales que intervienen en este bucle son Vo_res e Ii_res como principales, y las señales Vo_inc e Ii_inc que se redimensionan en las señales Vo_inc21 e Ii_inc21 como se mostraba en Figura 14. Esto se hace para ajustar estas señales a la resolución final y no introducir en la ALU (Unidad Aritmético Lógica) encargada de realizar la suma, más bits de los que se tendrán en cuenta en la redimensión como se muestra en el Código 5.

Como ya se ha explicado en el punto 3.3, el número de bits necesarios para representar un valor típico de la tensión de salida, Vo_res, sería Q10.14. Mientras que para el caso de la corriente de entrada, un tamaño correcto para representar un valor típico sería Q3.13. En ese mismo apartado se adelantó que serían suficientes 8 bits extra para representar estas señales que es lo que se va a demostrar en este capítulo. Para llegar a esta conclusión se ha comparado el modelo en coma fija con distintos tamaños para estas señales con el modelo *real*.

Para comprobar ambos modelos se hicieron tres pruebas distintas, ya que en circunstancias distintas pueden pasar cosas distintas y lo que para una señal en unas condiciones puede ser suficiente resolución, no lo es tanto para la misma en otras condiciones. Esto se puede ver claramente en la Tabla 10 en las figuras Figura 40, Figura 42 y Figura 44 donde se muestra la simulación resultante de los tres tipos de pruebas realizados para 4 bits de resolución extra:

- En primer lugar se realizó la simulación para el convertidor iniciado en su estado de equilibrio, es decir, sin transitorio, iniciando en 400 V la tensión de salida y en 1,5 A la corriente de entrada. Un ejemplo de esta simulación se puede ver en Figura 33.

- La siguiente prueba que se realizó consistía en observar el transitorio de corriente, para lo que se inició la tensión de salida en 380 V con la corriente de entrada a 0 A. Este transitorio se puede apreciar en la Figura 35.

- La última prueba consistía en mantener la resistencia de salida a un valor fijo durante 2 ms y posteriormente aumentar este valor al doble. Con lo que se conseguía ver un transitorio en los valores tanto de tensión de salida como de corriente de entrada. La simulación resultado de estas pruebas se ve en la Figura 37.

Como se muestra en las tablas Tabla 9 y Tabla 10 las formas de onda en el caso de 0 bits de resolución extra ($N_i = N_v = 0$) son muy diferentes del modelo *real* que se toma como modelo patrón por su gran resolución. Para el caso de 8 bits extra se puede ver su gran parecido con la forma de onda del modelo *real*. Pero en el caso de 4 bits, para la simulación sin transitorio se ve una forma de onda distinta que no es tan acentuada en las otras simulaciones. Ante la imposibilidad de garantizar visualmente el error cometido es necesario tomar datos numéricos.

Tras realizar los tres tipos de pruebas anteriores se tomaron numerosos valores de las gráficas representadas con los que se hallaron los errores de tensión y corriente para distintas resoluciones, es decir, para distinto número extra de bits, respecto al modelo *real*. Por último se toma el máximo de los errores de tensión y corriente globales, es decir el máximo de corriente tras realizar las tres pruebas y el máximo de tensión de las mismas. De estos máximos se obtiene la Tabla 8.

Los datos de la Tabla 8 representan los errores máximos de corriente y de tensión que se dan en las señales de salida del convertidor conmutado. Para poder tener un buen criterio para determinar si un error es o no grande, es necesario conocer que los valores típicos de la tensión de salida y de la corriente de entrada. Para la tensión de salida un valor típico puede ser de 400 V, mientras que para la corriente de entrada su valor medio es de 1,5 A. Con esta aclaración se puede determinar si los errores mostrados en la Tabla 8 son grandes o no.

N_v	N_i	Formato Ii	Formato Vo	Error máximo Vo	Error máximo Ii
0	0	Q3.13	Q10.14	67,818 V	4,464952 A
2	8	Q3.15	Q10.22	0,796 V	0,177643 A
3	8	Q3.16	Q10.22	0,182 V	0,031570 A
4	4	Q3.17	Q10.18	3,283 V	0,187910 A
4	8	Q3.17	Q10.22	0,142 V	0,029190 A
5	8	Q3.18	Q10.22	0,125 V	0,011470 A
6	6	Q3.19	Q10.20	0,802 V	0,168560 A
6	8	Q3.19	Q10.22	0,094 V	0,006470 A
7	7	Q3.20	Q10.21	0,528 V	0,035230 A
8	7	Q3.21	Q10.21	0,506 V	0,034820 A
7	8	Q3.20	Q10.22	0,103 V	0,006650 A
8	8	Q3.21	Q10.22	0,090 V	0,006370 A
16	16	Q3.29	Q10.30	0,061 V	0,005670 A

Tabla 8. Tabla con los errores máximos de tensión y corriente para distintas resoluciones de Vo_res y de Ii_res

En la Tabla 8 se observan varios puntos clave para la decisión de la resolución de las señales Vo_res e Ii_res. A continuación se profundiza en la información que aporta cada uno de esos puntos importantes de la Tabla 8:

- En primer lugar, si no se añaden bits extra, es decir, se emplean los formatos Q3.13 y Q3.14 para Ii y para Vo respectivamente, la evolución de la señal difiere mucho de la real con errores de hasta 67,818 V en tensión y de 4,464952 A en corriente, pero no solo eso, sino que como se observa en la Tabla 9, la forma de onda es muy distinta a la real. Esto se da tanto para la simulación con transitorio, como para la que no tiene transitorio y para la simulación con resistencia de salida variable.

- Por otro lado, la resolución con 16 bits extra no es muy diferente a la de 8 bits, por lo tanto el compromiso entre el número de bits añadidos y la mejora conseguida no es crítico en este salto y por ello no se realizan simulaciones intermedias.

- Otro de los puntos críticos que se aprecian en la tabla 8 es que la resolución de Ii no debe ser menor de 8 bits, pues cuando esto ocurre aumentan las pérdidas de Vo más de 0,5 V. Es decir según disminuye el número de bits en Ii, aunque aumente el número de bits para Vo, no es suficiente para mejorar significativamente el error. Esto se aprecia claramente en el caso de 8 bits para Vo y 7 bits para Ii.

- Por último, si se mantiene el número de bits para Ii a 8 y se disminuye el número de bits en Vo, no se aprecian pérdidas muy significativas hasta que se reduce el número de bits para Vo a 2. Para este caso se dan unas pérdidas mayores de 0,5 V en Vo y mayores a 0,1 A para Ii. Con esto último se aprecia que es necesaria mayor resolución para la corriente de entrada que para la tensión de salida del convertidor.

Por lo tanto, tal y como se ve en la Tabla 8, hay resoluciones que aportan un error similar y todas ellas se podrían barajar para la elección de la resolución oportuna. Entre estas resoluciones existen algunas cuyo error es permisible. Estos formatos son el de resolución de 16 y 8 bits extra para ambas señales, además de 8 bits para Ii con resoluciones de Vo desde 3 bits hasta 7. Posteriormente, pese a que el modelo de 16 bits de resolución extra obviamente logra la mejor resolución, se tendrán en cuenta los recursos empleados tras llevar a cabo cada uno de estos modelos en FPGA y así tomar una decisión en cuanto al tamaño apropiado para las señales.

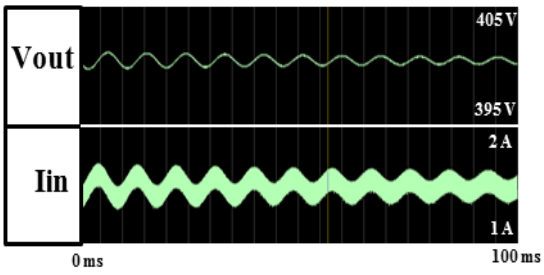	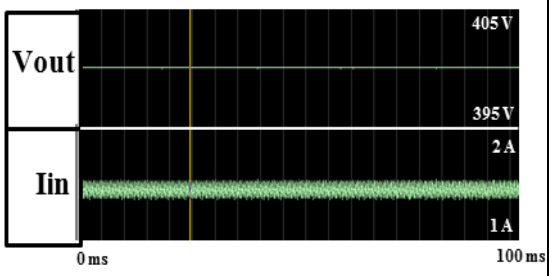
Figura 33. Simulación en lazo abierto del boost real sin transitorio	**Figura 34. Simulación en lazo abierto del boost con 0 bits extra sin transitorio**
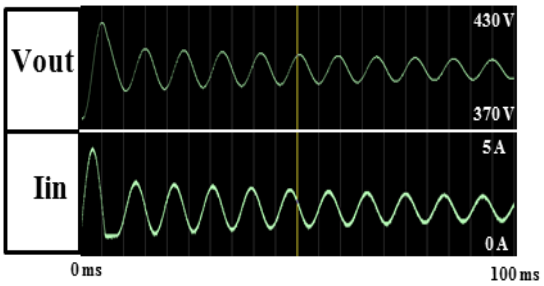	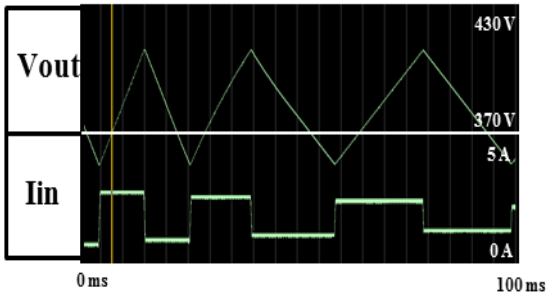
Figura 35. Simulación en lazo abieto del boost real con transitorio	**Figura 36. Simulación en lazo abierto del boost con 0 bits extra con transitorio**
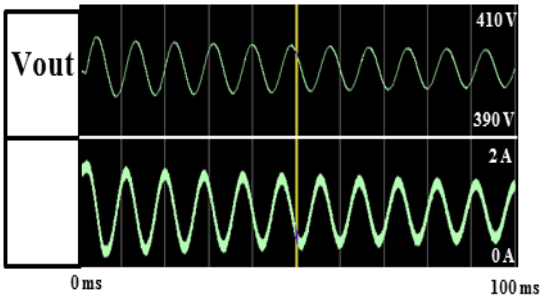	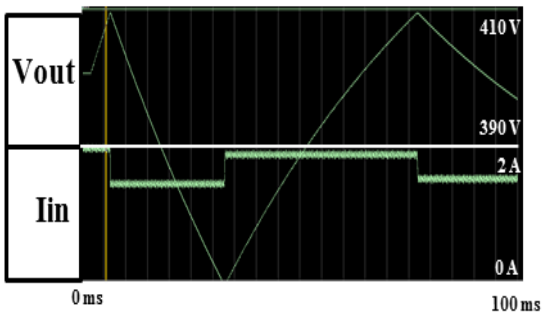
Figura 37. Simulación en lazo abierto del boost real con resistencia de salida variable	**Figura 38. Simulación en lazo abierto del boost con 0 bits extra con resistencia variable**

Tabla 9. Comparación visual del modelo real del boost con el modelo en coma fija de 0 bits extra en lazo abierto

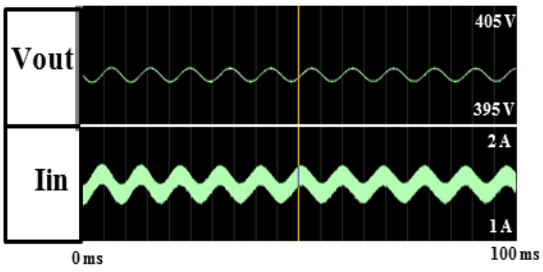

Figura 39. Simulación en lazo abierto del boost con 8 bits extra sin transitorio

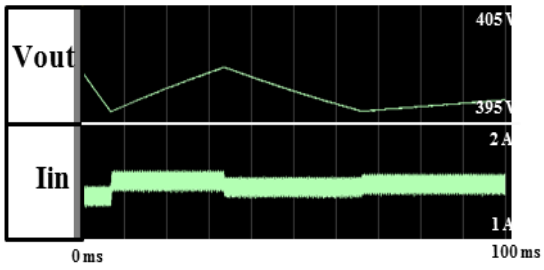

Figura 40. Simulación en lazo abierto del boost con 4 bits extra sin transitorio

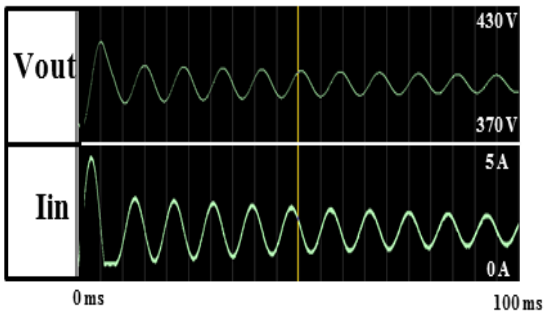

Figura 41. Simulación en lazo abierto del boost con 8 bits extra con transitorio

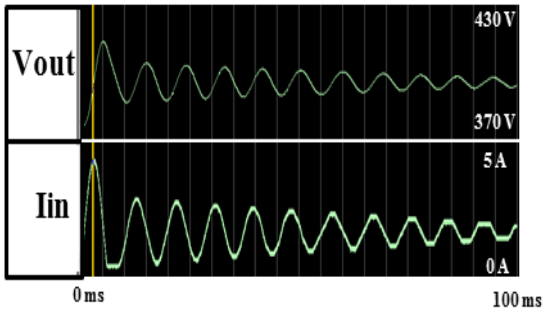

Figura 42. Simulación en lazo abierto del boost con 4 bits extra con transitorio

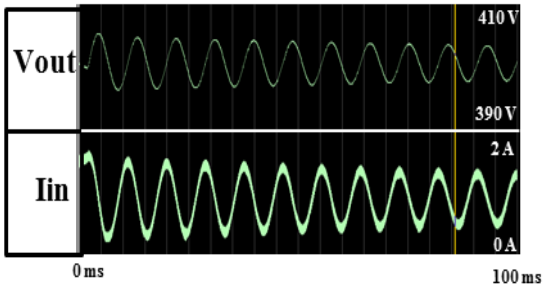

Figura 43. Simulación en lazo abierto del boost con 8 bits extra con resistencia variable

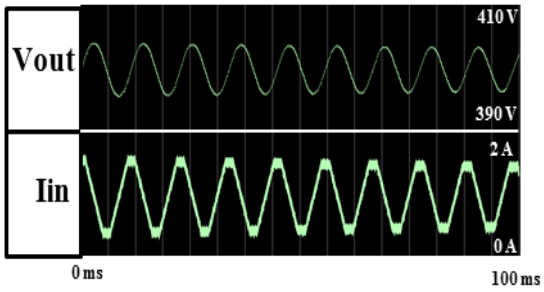

Figura 44. Simulación en lazo abierto del boost con 4 bits extra con resistencia variable

Tabla 10. Comparación visual del modelo en coma fija de 8 y 4 bits extra del boost en lazo abierto

Como se ha comentado anteriormente, en el modelo en lazo cerrado no se aprecia la diferencia entre un modelo correcto y un modelo con errores. Para evidenciar más aun este hecho se ha realizado una nueva prueba. En primer lugar se ha realizado la simulación del modelo real en lazo cerrado sin pérdidas. De esta simulación no solo se toma el valor de tensión de salida y el de corriente de entrada, sino que además, se toma el valor de la conductancia que sirve como elemento que regula la tensión de salida. Esta conductancia es parte del regulador y varía su valor en función de la tensión de salida del instante anterior. Un valor típico para esta conductancia según a representación empleada en las tablas Tabla 11 y Tabla 12 sería 1600, que por su representación en el regulador como un Q-5.18 sería de 0,0061 Ohm.

Tras esta simulación del modelo *real*, se toman los mismos valores en los mismos instantes para el modelo en coma fija con 8 bits extra y posteriormente para 4 bits extra. Con todos los resultados se hallan los errores en cada instante de tiempo de las señales de salida del modelo real con cada uno de los modelos en coma fija que se muestran en las tablas Tabla 11 y Tabla 12, es decir, para el modelo que emplea 8 bits de resolución extra en Vo_res e Ii_res y para el modelo que emplea 4 bits extra para las mismas señales.

ERRORES MODELO SIN PÉRDIDAS CON 8 BITS EXTRA PARA Vo_res e Ii_res:						
Tiempo	2200 us	4770 us	7620 us	72310 us	74690 us	77460 us
error Vo	0,027	0,067	0,037	0,259	0,281	0,149
error Ii	0,00076	0,00274	0,00389	0,0012	0,00188	0,00227
error Gin	2	2	2	1	1	1

Tabla 11. Errores del modelo en coma fija de 8 bits en lazo cerrado para el modelo sin pérdidas

ERRORES MODELO SIN PÉRDIDAS CON 4 BITS EXTRA PARA Vo_res e Ii_res:						
Tiempo	2200 us	4770 us	7620 us	72310 us	74690 us	77460 us
error Vo	0,277	0,317	0,537	0,509	0,281	0,101
error Ii	0,00271	0,00078	0,00389	0,01096	0,01556	0,0075
error Gin	2	2	2	15	15	15

Tabla 12. Errores del modelo en coma fija de 4 bits en lazo cerrado para el modelo sin pérdidas

Como se puede ver en Tabla 11 y Tabla 12, los errores en tensión de salida son muy distintos pero no tanto en la corriente de entrada. La razón es que el lazo de corriente es rápido y consigue que el error de corriente no crezca tanto por su elevado ancho de banda. El lazo de tensión es más lento y puede llegar a haber mayores errores de Vout independientemente de lo bueno que sea el modelo. Por otro lado la conductancia varía mucho para el modelo con 4 bits. Esto se debe a que el lazo cerrado intenta llevar la tensión de salida a un valor de consigna aunque para ello deba variar la relación en el regulador. Por tanto, habría que juzgar la resolución de los modelos por el error en Gin y no en Vout o Iin. Sin embargo, puede que distintas causas de error provoquen

variaciones en Gin de distinto signo y se compensen. De ahí que la verificación del modelo en lazo cerrado no sea tan fiable.

4.1.2 Resolución para el modelo con pérdidas

En este caso, se debe ajustar la resolución de los elementos que representan las pérdidas del circuito, es decir, el formato en el que se definen Rl, Rm, ESR, Rd y Vd. Para determinar estos tamaños se realiza la simulación del modelo real con pérdidas y se compara con el modelo en coma fija con pérdidas en el que se varían los formatos en los que se almacenan las pérdidas.

Por lo tanto se simulan ambos casos, modelo real y modelo en coma fija, fijando un formato para Vo y para Ii de Q10.22 y Q3.21 respectivamente, es decir, con 8 bits de resolución extra que como se ha comprobado para el caso sin pérdidas es una buena aproximación. La simulación que se va a llevar a cabo para verificar este modelo es con transitorio, con 0 A para la corriente de entrada inicial y 380 V para la tensión de salida inicial. Tras esta simulación se toman varios datos para distintos tiempos para cada una de las resoluciones que se ve en la primera columna de la Tabla 13. Con estos datos se hallan los errores de tensión de salida y de corriente de entrada para cada una de las resoluciones respecto al modelo con aritmética *real*. En la Tabla 13 se muestran los errores máximos de tensión de salida y de corriente de entrada para cada una de las resoluciones.

Resolución	Error máximo Vo	Error máximo Ii
12 bits	0,008	0,00037
10 bits	0,010	0,0004
8 bits	0,014	0,00135
6 bits	0,059	0,00385
4 bits	0,171	0,01269
3 bits	0,779	0,04789

Tabla 13. Errores y recursos empleados para distitnas resoluciones en las pérdidas

Los errores provenientes de las diferentes reosluciones en las pérdidas no son muy trascendentales como se observa en la Tabla 13 para resoluciones de 6 o más bits. Ya que a partir de esta resolución, para menos bits, aparecen errores no despreciables. Por lo tanto se estima el uso de 8 bits mínimo para obtener errores admisibles. Ya que como se puede ver con más resolución, por ejemplo con 12 bits, los errores no disminuyen drásticamente, hay una diferencia de error de 0, 006 V, a diferencia de lo que ocurre cuando se disminuye la resolución de 8 bits hasta 4 bits, que hay una diferencia de 0,157 V.

Por otro lado, para comprobar que la elección de 8 bits de resolución extra para Vo y para Ii es acertada, se realiza una nueva comprobación. Para esta se toman 8 bits de

resolución para las pérdidas y se varían los bits extra de Vo y de Ii. Nuevamente esto se lleva a cabo para una simulación con transitorio de corriente y en la tabla se recopilan únicamente los errores máximos para cada resolución.

Resolución	**16 BITS**		**8 BITS**		**6 BITS**		**4 BITS**		**0 BITS**	
Formatos	**Vo**	**Ii**	**Vo**	**Ii**	**Vo**	**Ii**	**Vo**	**Ii**	**Vo**	**Ii**
	Q10.30	**Q3.29**	**Q10.22**	**Q3.21**	**Q10.20**	**Q3.19**	**Q10.18**	**Q3.17**	**Q10.14**	**Q3.13**
Vout	0,01500 V		0,01400 V		0,31600 V		0,71700 V		46,403 V	
Iin	0,00124 A		0,00135 A		0,03055 A		0,07881 A		2,8145 A	

Tabla 14. Error de Vo y para Ii del boost con pérdidas

Los resultados de esta comparativa se ven en la Tabla 14. En esta se muestra el error máximo de tensión y el error máximo de corriente para varias resoluciones. Determinando que aunque este error sea similar para 16 bits, la mejora añadida no es muy elevada en comparación con la resolución de 8 bits extra. Esto ya se ha visto anteriormente para el modelo sin pérdidas, en el que estas señales, Ii y Vo, tenían un comportamiento parecido para ambas resoluciones de bits. De igual manera, para 0 bits extra se da un comportamiento muy diferente del esperado incluso en las formas de onda al igual que ocurría en el modelo sin pérdidas.

Al igual que para el modelo sin pérdidas, para el modelo con pérdidas también se ha realizado una simulación en lazo cerrado de la que se extraen la tensión de salida, la corriente de entrada y la conductancia del regulador. Estos valores se han tomado del modelo real y del modelo en coma fija tanto para 8 como para 4 bits de resolución extra y se determinan los errores de los modelos en coma fija frente al modelo *real*. Estos datos se muestran en las tablas Tabla 15 y Tabla 16.

ERRORES MODELO CON PÉRDIDAS CON 8 BITS:						
Tiempo	2200 us	4770 us	7620 us	72310 us	74690 us	77460 us
error Vo	0,029	0,086	0,01	0,166	0,113	0,092
error Ii	0,00067	0,00091	0,00212	0,00154	0,00253	0,00062
error Gin	0	0	0	2	2	2

Tabla 15. Errores del modelo en coma fija de 8 bits en lazo cerrado para el modelo con pérdidas

ERRORES MODELO CON PÉRDIDAS CON 4 BITS:						
Tiempo	2200 us	4770 us	7620 us	72310 us	74690 us	77460 us
error Vo	0,279	0,164	0,26	0,416	0,113	0,158
error Ii	0,00262	0,00287	0,00212	0,01521	0,02792	0,01429
error Gin	0	0	0	18	18	18

Tabla 16. Errores del modelo en coma fija de 4 bits en lazo cerrado para el modelo con pérdidas

Al igual que para el caso sin pérdidas, los errores de tensión son mayores para el caso de 4 bits de resolución extra pero no ocurre lo mismo con la corriente de entrada debido a que el lazo de corriente es más rápido y no permite que el error crezca tanto como en la tensión dónde el lazo de es más lento y puede dar lugar a errores mayores. Se podría juzgar el modelo por la Gin, pero como se ha explicado para el caso sin pérdidas, esta conductancia puede estar influenciada por otras causas que provoquen la desaparición de errores grandes. Por lo tanto, se puede comprobar nuevamente que la verificación en lazo cerrado no es tan fiable.

4.2. Área y frecuencia

Además de la importancia de la resolución, hay que tener en cuenta que si aumenta el número de bits empleados, pese a que mejore la resolución, puede requerir el uso de más elementos de la FPGA en la que se verifique. En este apartado se analizarán los resultados de frecuencia y área que ocupa cada *boost* en una FPGA teniendo en cuenta las resoluciones obtenidas en el apartado anterior. Debido a que el modelo *real* no es emulable, solo se baraja la verificación en FPGA del modelo en coma fija, pero eso sí, para distintos formatos de las señales de Vo y de Ii además de las resoluciones de las pérdidas como ya se ha hecho en el apartado anterior. Esto es importante porque el compromiso entre la precisión obtenida y los recursos empleados que determina el número de bits utilizados será crítico para la decisión de los tamaños finales óptimos.

Como en el procedimiento para determinar la resolución, para conseguir el área y la frecuencia de cada modelo en coma fija, es decir para cada uno de los formatos que se barajan para diseñar el modelo del convertidor conmutado, se realizará primero el análisis para el modelo sin pérdidas y posteriormente para el modelo con pérdidas. Además para ambos casos se realiza el análisis del boost solo, no del modelo completo junto al control.

4.2.1 Área y frecuencia para el modelo sin pérdidas

Tras realizar la emulación del modelo sin pérdidas mediante el programa Xilinx obtenemos los recursos que se emplearán para llevar a cabo cada uno de los modelos sin pérdidas. Para los casos con mejores resoluciones, es decir con errores menores, se realiza la verificación mediante Xilinx. Con este programa se puede ver el número de Flip Flops (FF), el número de LUTs (Look-Up Table) y el número de multiplicadores empleados, además del periodo del *boost* para cada caso.

Para el modelo sin pérdidas, se verifican los recursos ocupados y la frecuencia necesaria dependiendo del número de bits destinados a las señales Vo_res e Ii_res. Esta verificación se muestra en la Tabla 17.

Con los resultados de la Tabla 17 y de la Tabla 8, se determina que el uso de la resolución de 8 bits extra para la corriente de entrada y 8 para la tensión de salida son el mínimo número de bits extra recomendable. Los recursos empleados para este caso de 8 bits no son excesivos respecto al resto de opciones y el periodo de reloj tiene un valor similar para todos los modelos. Por lo tanto, las señales involucradas en este caso, Vo_res e Ii_res, tendrán un formato final de Q10.22 y Q3.21 respectivamente. Por otro lado, estos formatos implican las dimensiones para los incrementos Vo_inc21 e Ii_inc21, para las cuales los formatos son Q-9.22 y Q-7.21 respectivamente.

Bits extra Vo	Bits extra Ii	Nº FF	Nº LUTs	Nº Mult	Periodo (ns)
16	16	75	492	2	39,278
8	8	59	447	2	35,988
6	6	55	459	2	35,992
7	7	56	467	2	35,994
7	8	57	470	2	36,642
6	8	57	440	2	36,924
5	8	56	441	2	35,940
4	8	55	392	2	35,914
3	8	54	390	2	35,648

Tabla 17. Tabla con los recursos empleados para distintas resoluciones de Vo_res y de Ii_res para el *boost* sin pérdidas

4.2.2 Área y frecuencia para el modelo con pérdidas

Para el modelo con pérdidas también se realiza una verificación del modelo con pérdidas en coma fija del *boost* solo, con lo que se obtiene la cantidad de recursos empleados y el periodo de reloj que emplea este modelo. Estos resultados se muestran en la Tabla 18, en la que se puede ver la diferencia entre los recursos ocupados en función de la resolución de las pérdidas que se estime, es decir para los distintos tamaños de Rl, Rm, Rd, Vd y ESR. Siempre para una resolución extra de 8 bits para las variables de estado, es decir, $N_v = N_i = 8$ bits para todos los casos.

Resolución	Nº FF	Nº LUTs	Nº Mult	Periodo (ns)
12 bits	91	642	5	54,945
10 bits	91	633	5	54,800
8 bits	91	624	5	54,869
6 bits	91	617	5	54,818
4 bits	91	609	5	54,751
3 bits	91	607	5	54,898

Tabla 18. Tabla con los recursos empleados para distintas resoluciones de las pérdidas para el *boost*

Por otro lado con estos resultados, se determina que la resolución mínima que tiene un buen compromiso entre error máximo y recursos empleados para las pérdidas es de 8 bits, como ya se adelantó durante la explicación del diseño del modelo del convertidor conmutado elevador con pérdidas.

Por lo tanto para este modelo con pérdidas se emplearán 8 bits extra para las señales Vo_res e Ii_res, como se resolvió en el apartado de precisión y entre los tamaños para las pérdidas se emplean 8 bits de resolución ya que suponen el mínimo número de bits extra que aporta un buen compromiso entre área ocupada, tiempo de resolución y error en las señales.

4.3. Tiempos de simulación y de emulación

Como se comentó en un principio, el objetivo de estas pruebas es proponer un modelo con el que verificar una fuente conmutada con control digital, para lo que se crea un modelo de planta digitalizado al igual que el control. Con este sistema se puede ahorrar tiempo para su verificación, ya que esta se realiza de manera digital.

Por lo tanto los tiempos que se muestran a continuación se toman para simulaciones y verificaciones en lazo cerrado, teniendo en cuenta el control y la planta digitalizada. Además para el caso de coma fija, los análisis que se hacen son para los casos resueltos en los apartados anteriores, es decir 8 bits de resolución para las pérdidas y para el caso de las señales realimentadas Vo_res e Ii_res se emplean los formatos Q10.22 y Q3.21 respectivamente.

4.3.1 Tiempos de simulación

Lo primero que se ha realizado para todos los casos es una simulación en la que se verifica su correcto funcionamiento. Esta simulación puede ser más o menos larga dependiendo del tipo de datos que se empleen para su resolución. Por lo tanto en la Tabla 19 se pueden ver los tiempos de simulación necesarios para simular 200 ms dependiendo de la aritmética de las señales que definan el boost y de si se incluyen o no las pérdidas. Para hallar estos tiempos se ha empleado un cronómetro con el que medir el tiempo real en el que se lleva a cabo la simulación. Por lo tanto estos tiempos pueden tener un ligero error de medida debido a la acción humana para accionar el cronómetro.

	Simulación 200 ms
Modelo real sin pérdidas	1’ 56,6”
Modelo real con pérdidas	1’ 58,5”
Modelo en coma fija sin pérdidas	4’ 5”
Modelo en coma fija con pérdidas	7’ 35,7”

Tabla 19. Tiempos de simulación

Las señales de salida del convertidor conmutado y la señal que regula la fuente, Gin, se pueden ver en las siguientes imágenes, de la Figura 45 a la Figura 48. En estas figuras se ven los 200 ms de simulación para cada uno de los casos contemplados.

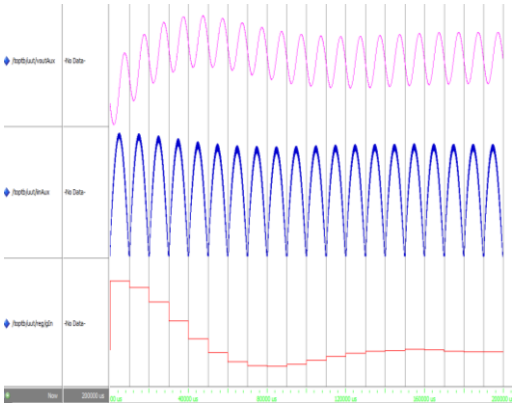

Figura 45. Simulación en lazo cerrado del modelo real sin pérdidas

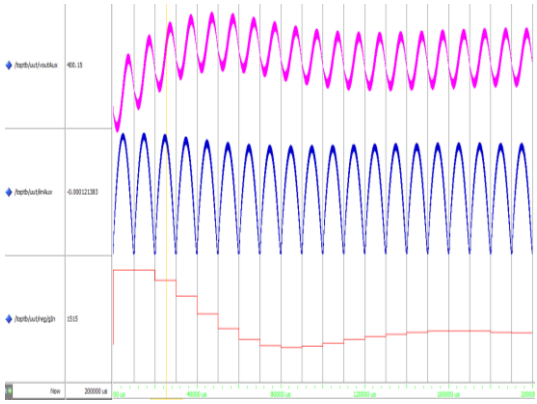

Figura 46. Simulación en lazo cerrado del modelo real con pérdidas

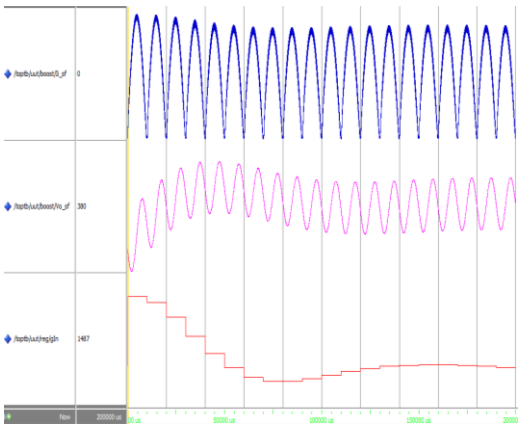

Figura 47. Simulación en lazo cerrado del modelo en coma fija sin pérdidas

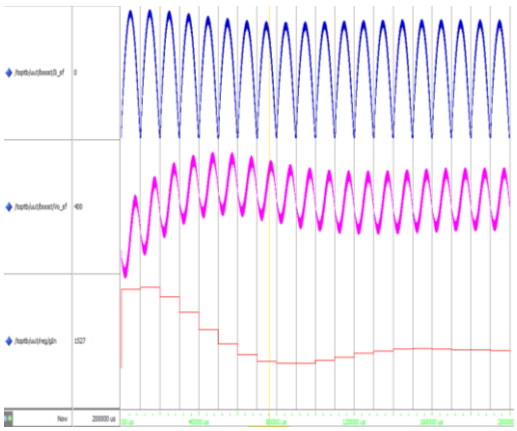

Figura 48. Simulación en lazo cerrado del modelo en coma fija con pérdidas

4.3.2 Tiempos de emulación

Para lograr unos tiempos de emulación adecuados, se debe ajustar el periodo de reloj requerido por el modelo en Xilinx mediante su especificación en el documente ucf. Este paso que ya se explicó en el punto 3.4 y es necesario para obtener unos tiempos de emulación verídicos. Ya que en este caso no se requiere el uso de ningún cronómetro, el tiempo que tarda en emular se halla a partir del periodo de reloj necesario que se resuelve directamente con Xilinx y que está relacionado con la especificación del ucf.

Originalmente, el periodo de reloj del circuito era de 10 ns que se ha definido para las simulaciones, por lo que en 200 ms hay 20 000 000 de ciclos de reloj, por lo tanto se multiplicará este número de ciclos por el periodo de ciclo de reloj que estime la FPGA al que puede trabajar. Con esto se halla el tiempo que se emplea en hacer la emulación y que se comparará con el tiempo requerido por las simulaciones realizadas en el apartado 4.3.1.

	Periodo	**Simulación 200 ms**
Modelo en coma fija sin pérdidas	40,057 ns	801,14 ms
Modelo en coma fija con pérdidas	57,452 ns	1149,04 ms

Tabla 20. Tiempos de verificación

La Tabla 20 muestra los tiempos de verificación para la simulación de 200 ms. En el periodo se muestra el tiempo por ciclo de reloj que debe invertir la FPGA para simular el convertidor. Este no es el reloj propio de la placa de la FPGA sino el reloj dividido que permite que se lleve a cabo el diseño. Estos relojes para el modelo sin pérdidas se pueden ver en la Figura 49, donde se selecciona como periodo de reloj el marcado en negro.

	Met	Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
1	Yes	PATH "TS_D2_TO_T2_ila_pro_0_path" TIG	SETUP		6.524ns		0
2	Yes	PATH "TS_J2_TO_D2_ila_pro_0_path" TIG					
3	Yes	PATH "TS_J3_TO_D2_ila_pro_0_path" TIG					
4	Yes	PATH "TS_J4_TO_D2_ila_pro_0_path" TIG	MAXDELAY		5.790ns		0
5	Yes	TS_Clk = PERIOD TIMEGRP "Clk" 20 ns HIGH 50%	MINLOWP...	14.000ns	6.000ns	0	0
6	Yes	TS_Inst_Clk_Div_CLKDV_BUF = PERIOD TIMEGRP "Inst_Clk_Div_CLKDV_BUF" TS_Clk * 3 HIGH 50%	SETUP HOLD	19.943ns 0.723ns	40.057ns	0 0	0 0
7	Yes	TS_U_TO_U = MAXDELAY FROM TIMEGRP "U_CLK" TO TIMEGRP "U_CLK" 15 ns	SETUP	12.608ns	2.392ns	0	0

Figura 49. Periodos de reloj de sintetización del modelo sin pérdidas en lazo cerrado57

Esta rapidez de emulación ha sido testeada en este trabajo, ya que al probar los modelos con ChipScope se aprecia la casi instantaneidad en la muestra de los datos que no se apreciaba en la simulación de los mismos.

4.3.3 Resumen de tiempos

Con los resultados de los dos apartados anteriores, se muestra la Tabla 21 que contiene los tiempos de simulación y emulación para una fuente conmutada con control digital y planta digital en VHDL. Las simulaciones han sido realizadas con la herramienta Modelsim y las emulaciones se han hecho en una FPGA gracias al programa Xilinx. La simulación mixta que se hizo en [3], se realizó con SystemVision 5.7 de Mentor Graphics.

	Simulación/ Emulación	**Tiempo de procesamiento para simular 200 ms**	**Porcentaje de aceleración**
Simulación mixta	Simulación	2h 13’ 21,751”	X
Modelo real sin pérdidas	Simulación	1’ 56,6”	68,63·X
Modelo real con pérdidas	Simulación	1’ 58,5”	67,52·X
Modelo en coma fija sin pérdidas	Simulación	4’ 5”	32,66·X
Modelo en coma fija con pérdidas	Simulación	7’ 35,7”	17,56·X
Modelo en coma fija sin pérdidas	Emulación	0,80114”	9987,96·X
Modelo en coma fija con pérdidas	Emulación	1,14904”	6963,86·X

Tabla 21. Tiempos de simulación de 200 ms

Esta tabla culmina el objetivo final de este trabajo en el que se logra demostrar como un modelo íntegramente digital tiene velocidades de procesamiento mucho más rápidas en comparación con los modelos mixtos explicado en el estado del arte.

5 CONCLUSIONES

La verificación del control digital es muy importante ya que es necesario testearlo junto al sistema que regula. Estos sistemas suelen ser analógicos porque casi todo en el mundo real es analógico. Ante la necesidad de verificar plantas analógicas junto a los controles digitales que las regulan se emplean sistemas mixtos capaces de manipular conjuntamente señales digitales y analógicas.

Los sistemas de simulación mixta son muy lentos por lo que nace la necesidad de otro tipo de simulación con el que conseguir mejores tiempos. Se descarta la simulación íntegramente analógica puesto que se busca la validación del modelo final del control, que es digital. Además, si como en nuestro caso la planta es un convertidor de potencia, un error en el control durante las simulaciones se traduce en voltajes o corrientes desorbitadas que pueden acarrear daños materiales o incluso personales.

Por lo tanto para evitar riesgos y poder llevar a cabo la simulación en un tiempo menor se emplea una técnica conocida como HIL (Hardware In-the-Loop) que consiste en digitalizar el modelo de planta para simularlo junto al control que lo regula. Si además se introduce este diseño en un dispositivo tipo FPGA se consigue simular (emular en este caso) este sistema a una velocidad mayor que si se ejecuta sobre un ordenador.

Para llevar a cabo esta simulación o emulación se ha realizado la digitalización del modelo de un convertidor elevador, para el que se han convertido las ecuaciones diferenciales en ecuaciones en diferencias. Para diseñar este modelo se emplea el lenguaje VHDL y su resultado se verifica en lazo abierto, es decir se prueba para unos casos sencillos con los que se pueda determinar el correcto funcionamiento del convertidor conmutado en VHDL. Una vez que este modelo funciona correctamente, se pasa a verificar el modelo completo del control digital junto a la fuente también digital. Este diseño permite validar el control sin provocar daños en la placa por lo que se pueden realizar modificaciones sin temor a estropear la planta.

Durante la digitalización de la planta, en este caso un convertidor elevador, se han desarrollado dos modelos con diferentes aritméticas. El primer modelo consiste en un modelo literal del convertidor y posteriormente se ha desarrollado un modelo que incluye las pérdidas propias del convertidor. En cuanto a las aritméticas, se ha empleado un tipo numérico real que posee una amplia resolución y por lo tanto muy buena precisión. De hecho se considera el modelo patrón. La otra aritmética empleada es coma fija con la que se han hecho varias pruebas de resolución hasta determinar los tamaños necesarios para disminuir los errores de resolución.

Con este desarrollo se permite acelerar la verificación respecto a la simulación mixta. Si se realiza una simulación mediante un entorno como *Modelsim,* definiendo la planta con aritmética *real* y siempre teniendo en cuenta el modelo con pérdidas que es el más parecido al real, se pueden lograr aceleraciones de 67 veces respecto a la simulación mixta. Si es necesario simular más ciclos de reloj o realizar una batería de pruebas mayor, por lo que se requiera mayor aceleración, es posible obtener aceleraciones de más de 6000 veces respecto al tiempo empleado en la simulación mixta. Para esta aceleración se debe realizar un modelo en coma fija para llevarlo a su emulación sobre FPGA. Para realizar el modelo en coma fija se deben concretar los tamaños de las señales empleadas, que para la mayoría es suficiente con una resolución entre 8 y 12 bits. Pero esta decisión se puede complicar al definir los tamaños de señales realimentadas que deban representar valores muy grandes junto a incrementos pequeños. Para estas señales se puede emplear una resolución aceptable tomando el entero superior a $\log_2(\frac{X}{\Delta X}) + 8$ bits extra. Siendo X el entero máximo que se represente y ΔX un incremento típico y 8 bits se ha tomado, tras las pruebas, como la mínima resolución extra a partir de la cual el modelo funciona correctamente. Por otro lado, se determina que realizar el modelo en coma fija para su simulación no aporta aceleraciones significativas respecto a la simulación con tipo *real,* por lo que no se recomienda esta opción para acelerar el proceso de pruebas.

6 LÍNEAS FUTURAS

Tras la viabilidad del trabajo con HIL, se puede ampliar este concepto para otros sistemas en lazo cerrado desde otros convertidores conmutados hasta plantas hidráulicas, para permitir que se verifiquen en poco tiempo evitando riesgos al probar controles inadecuados para las plantas que modelan.

Otro de los pasos que se abre tras este trabajo, además de la realización de más modelos, consistiría en pasar el modelo descrito a un módulo independiente en FPGA. En este trabajo se ha introducido todo el sistema en la misma FPGA por lo que sólo se ha verificado el empleo del algoritmo de control sobre la planta que regula. Pero también sería necesario probar las señales de entrada y salida, es decir, se debería introducir el diseño en FPGA y junto a unos convertidores digitales a analógicos se reflejaría íntegramente el funcionamiento de una planta analógica. Es decir, sustituir la planta por una caja negra que se comporta igual, y que además tiene los mismos conectores con el controlador. Así se comprobaría tanto el algoritmo de control como el hardware completo que lo regula.

Además se pueden tener en cuenta las pérdidas de segundo orden que no han sido registradas en los diseños elaborados en este trabajo y así poder validar si su empleo es rentable frente a la complejidad que añaden en el diseño.

Por otro lado, existen HIL comerciales que tienen menor integración que el propuesto en este trabajo, pero que se pueden comparar con lo aquí desarrollado.

Para finalizar, el uso de controles digitales y su verificación mediante la digitalización de los modelos analógicos está en proceso de investigación e inserción en el mercado actual y gracias a las ventajas que ofrece se aventura que se seguirá esta dinámica en los próximos años.

APÉNDICE

Modelo del *Boost* Real Sin Pérdidas

```
-- ------------------------------------------------
--                 BoostRealSP                    --
--

--            Modelo Real Sin perdidas           --
--
--            Sandra Jurado Jabonero             --
--                          TFG                  --
------------------------------------------------

library IEEE, WORK;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;

entity BoostRealSP is
      port(
            --In
            Clk : in std_logic;
            Reset : in std_logic;
            Mosfet : in std_logic;     -- On = '1', off = '0'
            Vg : in real;              -- Senal de entrada a el boost
            Ir : in real;              -- Senal requerida por la salida
            -- Out
            Iin : out real;                  -- Senal leida a la entrada del
Boost
            Vout : out real                  -- Tension leida a la salida del
boost
      );
end BoostRealSP;

architecture Behavioral of BoostRealSP is

      constant dt: real := 10.0e-9;   -- CLK
      constant C : real := 100.0e-6;  -- 100uF
      constant L : real := 5.0e-3;    -- 5mH

      constant dtL : real := dt/L;
      constant dtC : real := dt/C;

      constant Iini : real := 0.0;
      constant Vini : real := 380.0;

      -- Senales de salida
      signal Ii: real := 0.0;
      signal Vo : real := 0.0;

      -- Variables de entorno
      signal ic: real := 0.0;
      signal vl: real := 0.0;

begin
      Iin <= Ii;
      Vout <= Vo;
```

```
	-- Proceso de conmutacion del mosfet
	switchMUX : process (Mosfet, Vg, Ir, Ii, Vo)
	begin
		if Mosfet = '1' then		-- mosfet = on
			vl <= Vg;
			ic <= -Ir;
		else					-- mosfet = off
			if Ii > 0.0 then	-- Diodo en CCM
				vl <= (Vg-Vo);
				ic <= (Ii-Ir);
			else				-- Diodo en DCM
				vl <= 0.0;		-- Ii=0
				ic <= -Ir;
			end if;
		end if;

	end process switchMUX;

	-- Proceso de asignacion con el reloj y reset
	Asignacion : process (Clk, Reset)
	begin
		if Reset = '1' then
			Vo <= Vini;
			Ii <= Iini;
		elsif rising_edge(Clk) then
			Vo <= Vo + ic*dtC;
			Ii <= Ii + vl*dtL;
		end if;
	end process Asignacion;


end Behavioral;
```

Modelo del *Boost* Real con pérdidas

```
-- ------------------------------------------------
--                         BoostRealCP
--
--               Modelo Real con perdidas
--

--                     Sandra Jurado Jabonero
--                             TFG

------------------------------------------------

library IEEE, WORK;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;


entity BoostRealCP is
      port(
            --In
            Clk : in std_logic;
            Reset : in std_logic;
            Mosfet : in std_logic;     -- On = '1', off = '0'
            Vg : in real;              -- Senal de entrada a el Boost
            Ir : in real;              -- Senal requerida por la salida
            -- Out
            Iin : out real;                  -- Senal leida a la entrada del
Boost
            Vout : out real                  -- Tension leida a la salida del
Boost
      );
end BoostRealCP;

architecture Behavioral of BoostRealCP is

      constant dt: real := 10.0e-9;   -- CLK
      constant C : real := 100.0e-6;  -- 100uF
      constant L : real := 5.0e-3;    -- 5mH

      -- Componenetes que incluyen perdidas
      constant Rl: real := 0.6965; -- 0.6965 Ohm
      constant Rm: real := 0.18; -- 0.18 Ohm
      constant Vd: real := 1.45; -- 1.45 V
      constant Rd: real := 0.0; -- 0 Ohm de momento
      constant ESR: real := 3.316; -- 3.316 Ohm


      constant dtC : real := dt/C;
      constant dtL : real := dt/L;

      constant Iini : real := 0.0;
      constant Vini : real := 380.0;

      signal Ii: real := 0.0;
      signal Vo : real := 0.0;
      signal Vc : real := 0.0;
      signal ic: real := 0.0;
      signal vl: real := 0.0;

begin
      Iin <= Ii;
      Vout <= Vo;
```

```
	switchMUX : process (Mosfet, Vg, Ir, Ii, Vo)
	begin
		if Mosfet = '1' then		-- mosfet = on
			vl <= Vg-Ii*(Rl+Rm);
			ic <= -Ir;
		else				-- mosfet = off
			if Ii > 0.0 then
				vl <= Vg-Ii*(Rl+Rd)-Vd-Vo;
				ic <= Ii-Ir;
			else
				vl <= 0.0;
				ic <= -Ir;
			end if;
		end if;

	end process switchMUX;

	Asignacion : process (Clk, Reset)
	begin
		if Reset = '1' then
			Vc <= Vini;
			Ii <= Iini;
			Vo <= Vini;
		elsif rising_edge(Clk) then
			Vc <= Vc + ic*dtC;
			Ii <= Ii + vl*dtL;
			Vo <= Vc + ic*ESR;
		end if;
	end process Asignacion;


end Behavioral;
```

Modelo del *Boost* en coma fija sin pérdidas

```
-- ------------------------------------------------
--                  BoostQXYSP                   --
--                                               --
--                                                --
--            Sandra Jurado Jabonero              --
--                       TFG                      --
-------------------------------------------------

library IEEE, WORK, ieee_proposed;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;
use ieee_proposed.fixed_float_types.all;
use ieee_proposed.fixed_pkg.all;


entity BoostQXYSP is
	port(
		--In
		Clk : in std_logic;
		Reset : in std_logic;
		Mosfet : in std_logic;
		Vg : in std_logic_vector(12 downto 0);       -- (Q9.3)
		Ir : in std_logic_vector(12 downto 0);       -- (Q3.9)
		-- Out
		Iin : out std_logic_vector(11 downto 0);    --  (Q3.9)
		Vout : out std_logic_vector(11 downto 0)    --  (Q10.2)
```

```
      );
end BoostQXYSP;

architecture Behavioral of BoostQXYSP is

--  constant dt: real := 10.0e-9;
--  constant C : real := 100.0e-6;
--  constant L : real := 5.0e-3;
      ------------------------------------------------------------------------
-----------------------------------
      constant dtL : sfixed(-18 downto -35) := to_sfixed(0.000002, -18, -35);
-- Q-18.36 -- 18 bits
      constant dtC : sfixed(-13 downto -30) := to_sfixed(0.0001, -13, -30);
-- Q-13.31 -- 18 bits
------------------------------------------------------------------------------
      -- Puntos de inicio de la simulacion--
      constant Iini : sfixed(3 downto -21) := to_sfixed(0, 3, -21);
      constant Vini : sfixed(10 downto -22) := to_sfixed(380, 10, -22);

      -- Senales de entrada/salida en sfixed
      signal Vg_sf : sfixed(9 downto -3) := (others => '0');
      signal Ir_sf : sfixed(3 downto -9) := (others => '0');
      signal Ii_sf : sfixed(3 downto -9) := (others=>'0');
      signal Vo_sf : sfixed(10 downto -2) := (others=>'0');

      -- Senales de las variables de entorno
      -- Vout(k) = Vout(k-1)+ ic(k-1)/C*dt   --
      -- Iin(k) = Iin(k-1)+ vl(k-1)/L*dt     --
      signal ic: sfixed(3 downto -9) := (others=>'0');
      signal vl: sfixed(10 downto -3) := (others=>'0');

      -- Resultado de las senales--
      signal Ii_res: sfixed(3 downto -21) := (others=>'0');
      signal Vo_res : sfixed(10 downto -22) := (others=>'0');


      --Senales para asignal los 13 bits del QXY
      signal Iin_13: std_logic_vector(12 downto 0);
      signal Vout_13: std_logic_vector(12 downto 0);

      -- Incremento de las senales
      signal Ii_inc : sfixed(-7 downto -38) := (others => '0');
      signal Vo_inc : sfixed(-9 downto -39) := (others => '0');

      signal Ii_inc21 : sfixed(-7 downto -21) := (others => '0');
      signal Vo_inc21 : sfixed(-9 downto -22) := (others => '0');


begin

      -- Asignacion de senales entrada/salida a sfixed
      Vg_sf <= to_sfixed(Vg, Vg_sf);
      Ir_sf <= to_sfixed(Ir, Ir_sf);
      Vo_sf <= resize(Vo_res, Vo_sf);
      Ii_sf <= resize(Ii_res, Ii_sf);

      -- Paso a std_logic_vector de las salidas
      Iin_13 <= to_slv(Ii_sf); -- Paso a 13 bits con signo
      Vout_13 <= to_slv(Vo_sf);
      Iin <= Iin_13(11 downto 0) when Iin_13(12) = '0' else (others=>'0');
      Vout <= Vout_13(11 downto 0) when Vout_13(12) = '0' else (others=>'0');


      switchMUX : process (Mosfet, Vg_sf, Ir_sf, Ii_sf, Vo_sf, vl, ic)
      begin
            if Mosfet = '1' then
                  vl <= resize(Vg_sf,  vl);
                  ic <= resize((-Ir_sf), ic);
```

```
            else
                  if Ii_sf > to_sfixed(0, Ii_sf) then
                        vl <= resize((Vg_sf - Vo_sf), vl);
                        ic <= resize((Ii_sf - Ir_sf), ic);
                  else
                        vl <= to_sfixed(0, vl);
                        ic <= resize((-Ir_sf), ic);
                  end if;
            end if;

      end process switchMUX;

      Ii_inc <= vl*dtL;
      Ii_inc21 <= resize(Ii_inc, Ii_inc21);

      Vo_inc <= ic*dtC;
      Vo_inc21 <= resize(Vo_inc, Vo_inc21);

      Asignacion : process (Clk, Reset)
      begin
            if Reset = '1' then
                  Vo_res <= Vini;
                  Ii_res <= Iini;
            elsif rising_edge(Clk) then
                  Vo_res <= resize((Vo_res + Vo_inc21), Vo_res);
                  Ii_res <= resize((Ii_res + Ii_inc21), Ii_res);
            end if;
      end process Asignacion;



end Behavioral;
```

Modelo del *Boost* en coma fija con pérdidas

```
-- ------------------------------------------------
--                  BoostQXYCP                    --
--                                                --
--                                                --
--            Sandra Jurado Jabonero              --
--                      TFG                       --
------------------------------------------------

library IEEE, WORK, ieee_proposed;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;
use ieee_proposed.fixed_float_types.all;
use ieee_proposed.fixed_pkg.all;


entity BoostQXYCP is
      port(
            --In
            Clk : in std_logic;
            Reset : in std_logic;
            Mosfet : in std_logic;              -- On = '1', off = '0'
            Vg : in std_logic_vector(12 downto 0);  --(Q9.3)
            Ir : in std_logic_vector(12 downto 0);  --(Q3.9)
            -- Out
            Iin : out std_logic_vector(11 downto 0);  --(Q3.9)
```

```
            Vout : out std_logic_vector(11 downto 0)  -- (Q10.2)
      );
end BoostQXYCP;

architecture Behavioral of BoostQXYCP is

--  constant dt: real := 10.0e-9;
--  constant C : real := 100.0e-6;
--  constant L : real := 5.0e-3;
      ------------------------------------------------------------------------
      constant dtL : sfixed(-18 downto -35) := to_sfixed(0.000002, -18, -35);
-- Q-18.36 -- 18 bits
      constant dtC : sfixed(-13 downto -30) := to_sfixed(0.0001, -13, -30); --
Q-13.31 -- 18 bits
      ------------------------------------------------------------------------

      -- Componenetes que incluyen perdidas
      constant Rl:  sfixed(0 downto -8):= to_sfixed(0.6965, 0, -8); -- 0.6965
      constant Rm:  sfixed(0 downto -8):= to_sfixed(0.18, 0, -8); -- 0.18 =
      constant ESR: sfixed(2 downto -6) := to_sfixed(3.316, 2, -6); -- 3.316 =
      constant Vd:  sfixed(2 downto -6):= to_sfixed(1.45, 2, -6); -- 1.45
      constant Rd:  sfixed(2 downto -4):= to_sfixed(0.0, 2, -4); -- 0



      -- Puntos de inicio de la simulacion --
      constant Iini : sfixed(3 downto -17) := to_sfixed(0, 3, -17);
      constant Vini : sfixed(10 downto -18) := to_sfixed(380, 10, -18);

      -- Senales de entrada/salida en sfixed
      signal Vg_sf : sfixed(9 downto -3) := (others => '0');
      signal Ir_sf : sfixed(3 downto -9) := (others => '0');
      signal Ii_sf : sfixed(3 downto -9) := (others=>'0');
      signal Vo_sf : sfixed(10 downto -2) := (others=>'0');

      -- Senales de las variables de entorno
      -- Vout(k) = Vout(k-1)+ ic(k-1)/C*dt   --
      -- Iin(k) = Iin(k-1)+ vl(k-1)/L*dt     --
      signal ic: sfixed(3 downto -9) := (others=>'0');
      signal vl: sfixed(10 downto -7) := (others=>'0');


      -- Resultado de las senales --
      signal Ii_res: sfixed(3 downto -17) := (others=>'0');
      signal Vc_res : sfixed(10 downto -18) := (others=>'0');
      signal Vo_res : sfixed(10 downto -18) := (others=>'0');


      --Senales para asignal los 13 bits del QXY
      signal Iin_13: std_logic_vector(12 downto 0);
      signal Vout_13: std_logic_vector(12 downto 0);

      -- Incremento de las senales
      signal Ii_inc : sfixed(-7 downto -38) := (others => '0');
      signal Vc_inc : sfixed(-9 downto -39) := (others => '0');
      signal ESR_ic : sfixed(6 downto -15) := (others => '0');


      signal Ii_inc21 : sfixed(-7 downto -17) := (others => '0');
      signal Vc_inc21 : sfixed(-9 downto -18) := (others => '0');


      -- Auxiliares:
      signal aux1: sfixed(5 downto -17);
      signal aux2 : sfixed(7 downto -17);


begin
```

```
	-- Asignacion de senales entrada/salida a sfixed
	Vg_sf <= to_sfixed(Vg, Vg_sf);
	Ir_sf <= to_sfixed(Ir, Ir_sf);
	Vo_sf <= resize(Vo_res, Vo_sf);
	Ii_sf <= resize(Ii_res, Ii_sf);

	-- Paso a std_logic_vector de las salidas
	Iin_13 <= to_slv(Ii_sf); -- Paso a 13 bits con signo
	Vout_13 <= to_slv(Vo_sf);
	Iin <= Iin_13(11 downto 0) when Iin_13(12) = '0' else (others=>'0');
	Vout <= Vout_13(11 downto 0) when Vout_13(12) = '0' else (others=>'0');


	aux1 <= Ii_sf*(Rl+Rm);
	aux2 <= Ii_sf*(Rl+Rd);

	switchMUX : process (Mosfet, Vg_sf, Ir_sf, Ii_sf, Vo_sf)
	begin
		if Mosfet = '1' then
			vl <= resize((Vg_sf - aux1),  vl); -- Vg-Ii*(Rl+Rm)
			ic <= resize((-Ir_sf), ic);      -- Ir
		else
			if Ii_sf > to_sfixed(0, Ii_sf) then
				vl <= resize((Vg_sf - aux2 - Vo_sf - Vd), vl);
-- Vg-Ii*(Rl+Rd)-Vd-Vo
				ic <= resize((Ii_sf - Ir_sf), ic); -- Ii-Ir
			else
				vl <= to_sfixed(0, vl);
				ic <= resize((-Ir_sf), ic);
			end if;
		end if;

	end process switchMUX;

	Ii_inc <= vl*dtL;
	Ii_inc21 <= resize(Ii_inc, Ii_inc21);

	Vc_inc <= ic*dtC;
	Vc_inc21 <= resize(Vc_inc, Vc_inc21);

	ESR_ic <= ic*ESR;

	Asignacion : process (Clk, Reset)
	begin
		if Reset = '1' then
			Vo_res <= Vini;
			Vc_res <= Vini;
			Ii_res <= Iini;
		elsif rising_edge(Clk) then
			Vc_res <= resize((Vc_res + Vc_inc21), Vc_res);
			Ii_res <= resize((Ii_res + Ii_inc21), Ii_res);

			Vo_res <= resize((Vc_res+ESR_ic),Vo_res);
		end if;
	end process Asignacion;



end Behavioral;
```

Testbench del boost en lazo abierto para FPGA

```
--------------------------------------------------------------------------
--
-- Archivo: BoostTb.vhd
--
-- Simulacion del boost en lazo abierto, QX.Y
--------------------------------------------------------------------------
-- Autor: Sandra Jurado Jabonero
--------------------------------------------------------------------------
-- EPS - UAM
--------------------------------------------------------------------------

library IEEE, WORK, ieee_proposed;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;
use ieee_proposed.fixed_float_types.all;
use ieee_proposed.fixed_pkg.all;


entity BoostTb_FPGA is
	Port ( Clk : in std_logic;
	       Reset : in std_logic;
	       Captura : out std_logic;
	       VoutTop : out std_logic_vector(11 downto 0);
	       IinTop : out std_logic_vector(11 downto 0) );
end BoostTb_FPGA;


architecture Behavioral of BoostTb_FPGA is

component BoostQXYSP
  port(
	-- Input ports
	Clk : in std_logic;
	Reset : in std_logic;
	Mosfet : in std_logic;
	Vg : in std_logic_vector(12 downto 0);
	Ir : in std_logic_vector(12 downto 0);
	-- Output ports
	Iin : out std_logic_vector(11 downto 0);
	Vout : out std_logic_vector(11 downto 0)
	);
end component;

	COMPONENT Clk_Div
	PORT(
		CLKIN_IN : IN std_logic;
		CLKDV_OUT : OUT std_logic;
		CLKIN_IBUFG_OUT : OUT std_logic;
		CLK0_OUT : OUT std_logic;
		CLK2X_OUT : OUT std_logic
		);
	END COMPONENT;


	signal ir : sfixed(3 downto -9):= to_sfixed(0.75, 3, -9);
	signal vg : sfixed(9 downto -3) := to_sfixed(200.0, 9, -3);
	signal vo : sfixed(10 downto -2);

	signal s_mosfet : std_logic;
	signal s_vout : std_logic_vector(11 downto 0);

	-- Clock period definitions
```

```
	signal duty : integer; -- Duty cycle
	constant T_Gout: integer := 200000; -- 2 ms
	signal contador: integer Range 0 to 1000;
	signal contadorRes: integer Range 0 to 2000000; --Contador para la
resistencia variable


	-- Resistencia de salida constante:
	-- Conductancia = 1/533.33 = 0.001875
	signal Gout : sfixed(-9 downto -34) := to_sfixed(0.001875, -9, -34);

	signal clkDiv: std_logic;


begin

		uut: BoostQXYSP port map (
			Clk => clkDiv,
			Reset => Reset,
			Mosfet => s_mosfet,
			Vg => to_slv(vg),
			Ir => to_slv(ir),
			Iin => IinTop,
			Vout => s_vout);

		Inst_Clk_Div: Clk_Div PORT MAP(
			CLKIN_IN => Clk,
			CLKDV_OUT => clkDiv,
			CLKIN_IBUFG_OUT => open,
			CLK0_OUT => open,
			CLK2X_OUT => open
		);

		--Asignacion salida del top
		VoutTop <= s_vout;

		-- Entrada Vg
		vg <= to_sfixed(200.0, 9, -3);

		-- Entrada Ir
		vo <= to_sfixed('0'&s_vout, vo);
		ir <= resize(vo*Gout, ir);

		CountProcess: process(clkDiv, Reset)
		begin
			if Reset = '1' then
				contador <= 0;
			elsif clkDiv ='1' and clkDiv'event then
				if contador < 999 then
					contador <= contador + 1;
				else
					contador <= 0;
				end if;
			end if;
		end process;


		CountProcess2: process(clkDiv, Reset)
		begin
			if Reset = '1' then
				contadorRes <= 0;
			elsif clkDiv ='1' and clkDiv'event then
				if contadorRes < T_Gout then
					contadorRes <= contadorRes + 1;
				else
					contadorRes <= T_Gout;
				end if;
			end if;
```

```
        end process;


        duty <= 500; -- ciclo de trabajo fijo

        MsftProcess: process(contador, duty)     -- Proceso de Mosfet
        begin
               if contador < duty then
                      s_mosfet <= '1';
               else
                      s_mosfet <= '0';
               end if;
        end process MsftProcess;

        CaptProcess: process(clkDiv, Reset)
        begin
               if Reset = '1' then
                      Captura <= '0';
               elsif clkDiv ='1' and clkDiv'event then
                      if contador = 999 then
                             Captura <= '1';
                      else
                             Captura <= '0';
                      end if;
               end if;
        end process CaptProcess;


    RoutProcess: process(contador)
        begin
        --Gout <= to_sfixed(0.001875, -9, -34);

        -- Valor variables de Gout en escalon --
        if contadorRes < T_Gout then
               Gout <= to_sfixed(0.001875, -9, -34);
        else
               Gout <= to_sfixed(0.0009375, -9, -34);
        end if;

        end process RoutProcess;



end Behavioral;
```

Testbench para el lazo cerrado para FPGA

```
--------------------------------------------------------------------------
--
-- Archivo: BoostTb.vhd
--
-- Simulacion del boost en lazo abierto, QX.Y
--------------------------------------------------------------------------
-- Autor: Sandra Jurado Jabonero
--------------------------------------------------------------------------
-- EPS - UAM
--------------------------------------------------------------------------

library IEEE, WORK, ieee_proposed;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.math_real.all;
use ieee_proposed.fixed_float_types.all;
use ieee_proposed.fixed_pkg.all;


entity BoostHW_tb is
	Port ( Clk : in std_logic;
			Reset : in std_logic;
			Ir : in std_logic_vector(11 downto 0);
			VoutRef : in std_logic_vector(11 downto 0);
			Captura : out std_logic;
			VoutTop : out std_logic_vector(11 downto 0);
			IinTop : out std_logic_vector(11 downto 0) );
end BoostHW_tb;


architecture Behavioral of BoostHW_tb is

component Top
	port (
		Clk : in std_logic;
		Reset : in std_logic;
		Ir : in std_logic_vector(11 downto 0);
		VoutRef : in std_logic_vector(11 downto 0);
		--Outputs of the ADCs
		Vout : out std_logic_vector(11 downto 0);
		Iin : out std_logic_vector(11 downto 0)
	);
end component;

COMPONENT Clk_Div
	PORT(
		CLKIN_IN : IN std_logic;
		CLKDV_OUT : OUT std_logic;
		CLKIN_IBUFG_OUT : OUT std_logic;
		CLK0_OUT : OUT std_logic;
		CLK2X_OUT : OUT std_logic
		);
END COMPONENT;

	signal clkDiv: std_logic;
	signal contador: integer Range 0 to 1000;

	signal s_ir : std_logic_vector(11 downto 0) := "000110000000"; -- 0.75 A
	signal s_voutRef : std_logic_vector(11 downto 0) := "011001000000"; --
400 V


begin
```

```
        uut: Top port map (
               Clk => clkDiv,
               Reset => Reset,
               Ir => s_ir,
               VoutRef => s_voutRef,
               Iin => IinTop,
               Vout => VoutTop);

        Inst_Clk_Div: Clk_Div PORT MAP(
               CLKIN_IN => Clk,
               CLKDV_OUT => clkDiv,
               CLKIN_IBUFG_OUT => open,
               CLK0_OUT => open,
               CLK2X_OUT => open
        );



        CountProcess: process(clkDiv, Reset)
        begin
               if Reset = '1' then
                      contador <= 0;
               elsif clkDiv ='1' and clkDiv'event then
                      if contador < 999 then
                             contador <= contador + 1;
                      else
                             contador <= 0;
                      end if;
               end if;
        end process;



    -- Captura un dato de cada 1000 --
    CaptProcess: process(ClkDiv, Reset)
           begin
               if Reset = '1' then
                      Captura <= '0';
               elsif ClkDiv ='1' and ClkDiv'event then
                      if contador = 999 then
                             Captura <= '1';
                      else
                             Captura <= '0';
                      end if;
               end if;
           end process CaptProcess;




    end Behavioral;
```

BIBLIOGRAFÍA

[1] Alberto Sánchez, Ángel de Castro y Javier Garrido, "A Comparison of Simulation and Hardware-In-The-Loop Alternatives for Digital Control of Power Converters". En IEEE Transactions on industrial informatics, vol 8 NO 3, Agosto 2012.

[2] Ángel de Castro. *Aplicación del control digital basado en hardware específico para convertidores de potencia conmutados*. Tesis doctoral, Universidad Politécnica de Madrid.

[3] Alberto Sánchez, *Aportaciones mediante implementación basada en sistemas embebidos al control digital de convertidores conmutados.* Tesis Doctoral, Universidad Autónoma de Madrid, Junio 2013.

[4] L. Barragan, I. Urriza, D. Navarro, J. Artigas, J. Acero, and J. Burdio, "*Comparing simulation alternatives of fpga-based controllers for switching converters*" in Industrial Electronics, 2007. ISIE 2007.Junio 2007.

[5] A. de Castro, T. Riesgo, O. Garcia, and R. Prieto, "*Comparing vhdl and vhdl-ams for modelling and simulation of power converters with digital control*" in XVIII Conference on Design of Circuits and Integrated Systems (DCIS), Noviembre 2003.

[6] Alejandro García Talón, *Control digital de fuentes de alimentación.* Proyecto fin de carrera, Universidad Autónoma de Madrid, Abril 2009.

[7] P. Zumel, M. García-Valderas, A. Lázaro, C. Loópez-Ongil, and A. Barrado, "*Co-simulation psim-modelsim oriented to digitally controlled switching power Converters*" in Control and Modeling for Power Electronics (COMPEL), 2010 IEEE 12th Workshop, Junio 2010.

[8] Daniel W. Hart, *Electrónica de Potencia*. Editorial: Prentice Hall. ISBN: 84-205-3179-0.

[9] David Bishop. *Fixed point package user's guide*. Guia para el usuario de un tipo sfixed. Paquetes VHDL-2008.

[10] Robert W. Erickson y Dragan Maksimovic. *Fundamentals of Power Electronics.* ISBN-10: 0792372700 - 13 ISBN: 978-0792372707, Enero 2001, 2 edición.

[11] A. Prodic and D. Maksimovic, *Mixed-signal simulation of digitally controlled switching converters* in Computers in Power Electronics, 2002. Proceedings. 2002 IEEE Workshop, Junio 2002.

[12] Xilinx. *Spartan-3E FPGA Family Data Sheet.* Hoja de datos de la FPGA Spartan 3, DS312. Julio 2013.